MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN SOFTWARE ENGINEERING

Principes de conception et de développement des interactions similaires

Application aux cockpits interactifs d'aéronefs

Della Pasqua, Sabrina

Award date:
2018

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2017-2018

Principes de conception et de développement des interactions similaires : Application aux cockpits interactifs d'aéronefs

Sabrina DELLA PASQUA

Maître de stage : Philippe PALANQUE

Promoteur : _____ (Signature pour approbation du dépot - REE art. 40)
Bruno DUMAS

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Remerciements

Mes remerciements s'adressent en premier lieu à mon maître de stage, Philippe PALANQUE, pour sa confiance et ses conseils qui m'ont permis de progresser sans cesse durant ces 4 mois de stage.

En effet, c'est grâce aux missions que l'on m'a confiées que j'ai pu m'améliorer.

Je remercie également Arnaud HAMON pour son aide et sa collaboration dans ce travail.

Je tiens également à remercier mon professeur, Bruno DUMAS, qui m'a suivi tout au long de ce stage, pour sa disponibilité et son aide si précieuse.

J'exprime également ma gratitude à l'égard de l'ensemble de l'équipe pour leur sympathie et la bonne ambiance qui ont favorisées mon intégration.

Enfin, je tiens surtout à remercier tous mes proches qui m'ont soutenus pendant la durée du stage et l'écriture de ce mémoire.

Résumé

Depuis peu, un nouveau concept fait sensation, dans le domaine de l'aviation, le cockpit interactif. Seulement, le cockpit interactif amène beaucoup de questions quant à la sécurité que nous devons garder intacte et pareille à la sécurité présente actuellement. Mais il est aussi sujet à la discussion quant à la ressemblance, la similarité qu'il a par rapport au cockpit physique. L'objectif de ce mémoire est d'étudier les similarités ou dissimilarités qu'il existe entre le cockpit actuel, qui est physique, et le futur cockpit, qui est interactif et, dans notre cas, tactile. Cette étude a été accomplie au travers de la modélisation de tâches et du prototypage afin de mesurer le niveau de similarité entre deux systèmes. Finalement, ce travail a pour objectif d'amener des pistes pour de futurs travaux à ce sujet ainsi qu'une piste de méthodologie à toute personne souhaitant entreprendre le développement d'une interaction similaire.

Abstract

Recently, a new concept is causing a sensation, in the field of aviation, the interactive cockpit. But the interactive cockpit brings many questions about the security that we must keep intact and similar to the security level currently present in the cockpit. But he is also subject to discussion about the similarity. It must be as close as possible. The purpose of this thesis is to study the similarities or dissimilarities existing between the current cockpit, which is physical, and the future cockpit, which is interactive and, in our case, tactile. This study was accomplished through task modeling and prototyping to measure the level of similarity between two systems. Finally, this work aims to bring some ideas for future works on this subject as well as a methodology track to anyone wishing to undertake the development of a similar interaction.

Table des matières

Table des figures

Première partie

Introduction

Chapitre 1

Introduction

Sommaire

"[..]Cockpit moderne et efficace qui intègre une large zone d'affichage associée à plusieurs systèmes de commande reconfigurables. Cette organisation conviviale s'accompagne d'écrans tactiles multiples, assurant à l'équipage une interaction intuitive englobant tous les systèmes avion et autres fonctions associées.[..]" [1] voici la manière dont Thales voit les futurs cockpits.

Ce mémoire part de cette idée de cockpit interactif du futur qui inspire les chercheurs depuis des années déjà. L'objectif général est de réduire la charge de travail, l'échange d'informations devenant de plus en plus important. Cette vision de futur cockpit impose la redéfinition de l'interaction homme-machine.

"Un cockpit conçu autour d'une interaction plus transparente entre le pilote et l'électronique n'est plus un concept purement intellectuel, mais une application viable qui, avec le nombre croissant de fonctionnalités et de tâches ajoutées à la charge de travail du pilote, deviendra essentielle pour l'avenir du transport aérien."

1 Contexte général

Le but de ce travail est d'étudier les similarités ou dissimilarités qui apparaissent dans différents types d'interface d'un cockpit. Nous allons étudier le sujet sur deux types d'interfaces spécifiques : la physique, celle-ci existant déjà dans les cockpits actuels, et la tactile, n'existant que très peu ou pas du tout dans les cockpits.

L'idée est de voir comment on peut migrer d'une interface physique vers une interface tactile tout en gardant le même niveau d'interaction et une interface sensiblement ressemblante.

1. https ://www.thalesgroup.com/sites/default/files/asset/document/avionics_2020_fr.pdf

Le stage lié à ce mémoire a été effectué à l'université Toulouse-III-Paul-Sabatier, en France, au laboratoire de recherche de l'IRIT. Pendant les trois mois et demi du stage, le travail s'est fait dans l'équipe ICS (Interactive Critical Systems), sous la responsabilité de Philippe Palanque, qui concentre ses recherches dans le domaine des interactions homme-machine et plus particulièrement pour les systèmes critiques.

2 Questions de recherche

Différentes questions vont émerger de cette étude :

— Comment peut-on avoir un ressenti d'un bouton physique similaire à celui du tactile ?
— Quels sont les avantages et les inconvénients d'avoir un affichage, un visuel, une interaction plutôt qu'un(e) autre ?
— Comment s'assurer d'avoir autant ou moins d'erreurs de manipulation avec une interface qu'avec une autre ?

3 Plan du travail

Dans ce travail, nous avons une première partie, après cette introduction, qui est un état de l'art. Celle-ci contient un chapitre reprenant le cadre général du mémoire. Dans ce cadre, nous retrouvons le contexte du mémoire avec la description du cockpit tel qu'il est à l'heure actuelle ainsi que la définition d'une interaction tactile. Les sections, qui suivent ce contexte, reprennent un ensemble de concepts-clés par rapport au travail tel que le système critique, le système interactif, le concept de similarité ou encore le modèle de tâches. Pour chacun des concepts, nous aurons une définition ou une description du concept ainsi que des notions qui y sont liées.

Dans la seconde partie, l'objectif est de montrer la contribution apportée à propos du sujet au travers une étude de cas qui sera détaillée. Nous commencerons par détailler de manière informelle le cas d'étude choisi. Ensuite, le cas sera détaillé formellement avec un modèle de tâche qui sera expliqué entièrement. On introduira l'interaction de la GoPro, interaction qui nous a servi de base pour la suite du travail avec les prototypes. Nous montrons donc nos prototypes juste après avec les explications de comment on est arrivé à ces prototypes. Ensuite, pour chaque prototype, nous aurons un modèle de tâches associé qui sera expliqué. Enfin, une explication très brève donnera un moyen de vérifier la complétude de notre travail effectué auparavant.

Dans la troisième partie, nous avons un chapitre consacré à une application développée dans le cadre de ce mémoire. Celui-ci a pour but de présenter brièvement l'application ainsi que d'expliquer pourquoi cette application a été créée et à quel besoin celle-ci répond.

Dans la dernière partie, se trouvera une conclusion avec l'élicitation de ce qui pourrait être amélioré ou étudié dans un futur proche ou éloigné sur la thématique de ce travail.

4 Méthologie

Au vu du sujet du mémoire et de son domaine qui est, pour rappel, le domaine des avions civils, il était important de se renseigner sur celui-ci.

Étant donné le domaine, il faut prendre en considération les risques encourus lors de la conception et du développement d'applications pour ce genre de discipline. On peut voir ces explications à la section 2 du chapitre 2 de la partie "État de l'art". N'étant pas experte dans le domaine, il a fallu que je me renseigne au sujet de la composition générale d'un avion civile (section 1.1) ainsi que de son fonctionnement (section 1.1.2).

Afin de connaître un peu plus le fonctionnement d'un avion ainsi que les procédures de vol, le laboratoire m'a fourni le FCOM (Flight Crew Operating Manual) du Airbus A350, du Airbus A380 [2] et le "Flight Deck and Systems Briefing for Pilots Flight Deck and Systems Briefing for Pilots" [3] toujours du Airbus A350 et du Airbus A380 [4].

Ensuite, il a fallu se renseigner sur les différents concepts que l'on allait aborder dans le travail tel que les systèmes interactifs ainsi que le concept de similarité.

Á nouveau au vu de notre sujet, il était nécessaire approfondir sur le sujet de la similarité. Plusieurs recherches ont donc été faites sur ce qui existait déjà à ce propos. C'est à partir de là qu'on a remarqué que ce type de recherche était plutôt inédit. Cette similarité est étudiée plus loin dans le mémoire.

Pour étudier ce concept dans le cadre de l'aviation, il a fallu trouver un moyen de formaliser au maximum le travail. Vu que notre travail se base sur l'étude des actions de l'utilisateur pour atteindre un objectif, le modèle de tâches était un choix évident. Ce type de modélisation n'étant pas enseigné à l'UNamur, il a fallu un temps de recherche et d'apprentissage. Après ces recherches et cet apprentissage, il est apparu que le modèle de tâches permet de décrire ces tâches utilisateurs de manière précise et formelle. Á partir de ce choix et constat, après quelques recherches, plusieurs outils étaient possibles mais Hamsters semblait être le plus adapté pour les raisons que nous voyons en section 5.3 du chapitre 2 de la partie "État de l'art".

Après avoir appris la notation du modèle de tâches selon Hamsters, le but était de produire le modèle de tâche pour l'action générale d'extinction de feu moteur avec le panneau plafond existant. Ce modèle est passé par de nombreuses étapes. En effet, il a fallu valider ce modèle et pour cela nous avons travaillé de manière itérative. Á chaque étape, la question était de savoir est-ce qu'on a été assez loin dans la granularité du modèle pour pouvoir comparer la similarité avec notre future interaction ? Pour le savoir, il faut se rendre à la section 1.2 du chapitre 3 dans la partie "Contribution".

2. Trouvable sur internet à l'adresse suivante : http ://www.avsimrus.com/f/documents-16/fcom-airbus-a380-57563.html
3. Trouvable sur internet à l'adresse suivante : http ://www.smartcockpit.com/docs/a350-900-flight-deck-and-systems-briefing-for-pilots.pdf
4. Trouvable sur internet à l'adresse suivante : https ://fr.scribd.com/document/327781313/A380-Briefing-for-Pilots-Part

Par la suite, pour notre étude, il a fallu imaginer une nouvelle interaction afin de pouvoir comparer la similarité avec l'interaction de base. Pour cela, nous sommes partis d'une base qui est l'interaction de déverrouillage de la GoPro. Une analyse de cette interaction a été faite à la section 1.3 et à la section 1.4.1 au chapitre 3 toujours de la partie "Contribution".

Après avoir fait cette analyse, plusieurs étapes ont été nécessaires afin d'atteindre un niveau de similarité considérable par rapport à l'interaction de base. Toutes les étapes, concernant les prototypes, ne sont pas spécifiées dans ce mémoire cependant il est possible de les retrouver en annexe A. Afin de réaliser ces dernières, les principes [5] de base de la conception d'IHM ont été utilisés tels que la forme, la couleur, la taille, la localisation des objets ainsi que la police d'écriture. Certaines étapes sont de toute autre nature et servent à adapter l'interaction de la GoPro en une interaction plus proche de notre interaction avec le panneau plafond en changeant le sens de l'interaction.

Dans la continuité de ce travail, les trois prototypes ont été analysés au niveau de l'interaction. L'interaction de chaque prototype a été transformée en modèle de tâches. En fonction de chaque modèle de tâches, on peut estimer la similarité avec notre modèle de tâches de base en fonction de la granularité des actions, en fonction du nombre d'actions effectuées, ceci est observable à la section 1.5.

Pour aller encore plus loin, le dernier prototype a été développé et mis en correspondance avec le modèle de tâches correspondant. Grâce à un outil développé dans le laboratoire, il a été possible de faire correspondre le modèle de tâches avec le prototype réalisé. Ainsi, pour chaque action effectuée sur le prototype il y avait une tâche correspondante dans le modèle de tâches. Ceci permet de vérifier qu'aucune tâche n'a été omise.

Ce travail a contribué à la sortie d'un article scientifique qui se trouve en annexe C.

5. https ://perso.liris.cnrs.fr/stephanie.jean-daubias/enseignement/IHM/LifIHM-CM5-ErgoTheories.pdf

Deuxième partie

État de l'art

Chapitre 2

Cadre général

Sommaire

1 Contexte

Cette section a pour but d'expliquer le contexte dans lequel ce travail se situe.

En effet, il est important de savoir actuellement comment se compose un cockpit et comment est-il utilisé. La notion de modalité est élémentaire puisque dans notre sujet nous souhaitons passer de la modalité physique à la modalité tactile. Mais il est aussi fondamental de savoir ce qu'on entend par interaction tactile pour bien comprendre la suite du travail.

Notez bien que deux personnes se trouvent toujours dans le cockpit, il s'agit du pilote (Captain) et du co-pilote (First Officer ou Captain) mais ici pour une question de facilité, seul le terme de pilote sera employé.

1.1 Cockpit actuel

Pour commencer la présentation des cockpits actuels, voici une image qui représente un cockpit d'Airbus A380 [1].

FIGURE 2.1 – Cockpit d'un Airbus A380

Comme le montre la photo, le poste de pilotage suivant présente de nombreux éléments. Ceux-ci sont, de manière générale, physiques. C'est-à-dire que le pilote doit tourner les boutons, retirer des capots, appuyer sur les boutons, etc.

Quoiqu'il utilise, le pilote a toujours une perception tactile lorsqu'il utilise les boutons.

1. http ://www.airliners.net/photo/Airbus/Airbus-A380-841/957790
Il est possible de l'observer en vue 360 ici : http ://magazin.lufthansa.com/xx/en/fleet/airbus-a380-800-en/panorama-photo-cockpit-a380-800/

Par exemple, certains boutons du cockpit sont crantés. De ce fait, lorsque le pilote utilise ce type de bouton il ressent les crans qui sont en relief sur le bouton. Un autre exemple, un simple bouton "poussoir" procure un ressenti puisque, lorsque le pilote appuie dessus, il a le retour du bouton : une sensation de ressort, etc.

Certaines fonctionnalités sont également accessibles via le clavier et le trackball présents sur le centre du cockpit.

1.2 Interaction tactile

L'interaction tactile est une interaction entre humain et ordinateur/périphérique tactile.
Pour comprendre l'interaction tactile, il faut d'abord savoir comment l'humain fonctionne et perçoit les sensations de toucher.
D'après Loomis et Lederman [PL03], la perception tactile est une perception qui est provoquée par des stimulations cutanées uniquement. Toujours d'après Loomis et Lederman, elle est perceptible lorsqu'on se trouve dans une posture statique particulière.

Or dans notre cas, le pilote est en position assise tout le long du processus. Ce qui implique, selon leurs constatations, que le pilote perçoit la perception tactile. Le fait qu'il soit toujours assis signifie que la position est constante. De ce fait, l'interaction tactile est uniquement cutanée.

2 Système critique

Le système auquel nous nous intéressons dans ce mémoire fait partie de ce qui est appelé les systèmes critiques. Cette section a pour but de définir les systèmes critiques, de définir la sureté de fonctionnement et de voir les problèmes qui découlent de cette dernière [Cro17].

2.1 Description

Lorsque l'on développe un concept ou un logiciel dans le monde de l'avionique, il est important de garder à l'esprit qu'un système dans un avion est un système dit critique. Les sous-sections de cette section se basent principalement sur les références suivantes [HAR16], [NPHDP].

Un système critique, dans notre cas, sera vu comme [PB] :

Un système dont le coût de développement est moindre par rapport à un coût d'une défaillance potentielle du système.

C'est un système dont les défaillances peuvent entraîner des pertes de vies humaines, qui ont une grande valeur humaine et matérielle et/ou des dommages inadmissibles sur l'environnement.

Il y a plusieurs exemples de systèmes critiques dans les transports, la médecine, le secteur de l'énergie, ... mais celui qui nous intéresse est dans le domaine de l'aviation.

2.1.1 Classification

De par la nature de ce type de système, il y a deux approches possibles. Dans un premier temps, il y a l'approche zéro-défaut. Celle-ci a pour but de garantir que le système est "defect free". Il a été prouvé que c'est un bon moyen de détecter et de supprimer les défaillances, bugs au moment du développement, fautes "naturelles" (exemple : bit-flip dû à la radiation). Cette catégorie est basée sur des méthodes dites formelles.

Dans un second temps, il y a l'approche de tolérance à la faute. Ici, le principe est différent et il promouvoit :

La redondance : il existe plusieurs versions différentes du système.

La diversité : les différentes versions sont développées en utilisant des technologies différentes (plusieurs langages de programmation, des notations différentes pour décrire les spécifications, utilisé des périphériques d'entrée/sortie différents), des fournisseurs différents. Pour éliminer un maximum de "common point of failure" les interfaces utilisateurs doivent assurer la *diversité*. La diversité nécessite des interfaces utilisateurs bien différentes en matière de structure et de techniques d'interaction. Cependant, cela ne veut pas dire que le fond doit changer ; les différentes versions doivent effectuer les mêmes tâches et avoir les mêmes objectifs.

La ségrégation : les différentes versions sont intégrées dans l'environnement opérationnel par des moyens indépendants ; celle-ci permet entre autres de ne pas propager la défaillance sur d'autres versions (communément appelé "common point of failure").

2.1.1.1 Redondance

Dans notre cas, la redondance est fortement utilisée. Dans le cockpit, les systèmes ont, de manière générale, un système de back-up. Cela permet au pilote d'avoir plusieurs moyens afin d'effectuer une action particulière ou même d'avoir une même information affichée sur plusieurs écrans différents. On peut voir un exemple de cette méthode dans la figure 2.2.

FIGURE 2.2 – Dans le cockpit du A380, il y a deux moyens de contrôler le cap de l'avion : le premier en utilisant le FCU et le deuxième en utilisant l'application FCU (FCU backup) et le KCCU

Cette redondance peut aussi être faite à un niveau plus technique. Par exemple, les défaillances liées à la souris sont diminuées grâce à l'utilisation de plusieurs configurations "similaires" basées sur l'utilisation de plusieurs touches sur le clavier.

En terme de design, il est important que les interfaces redondantes ne soient pas dégradées par rapport à l'interface "d'origine".

2.2 Qualification des logiciels

Dans le cas particulier de l'aviation, plusieurs niveaux de criticités ont été mis en place pour qualifier les logiciels via la norme DO-178C [Pot12] [Ver] [I.S] [HAR16] [RTC12], les niveaux étant du A (le plus critique) au E (le moins critique) :

A : Catastrophiques - Si une erreur/panne survient à ce niveau, elle est qualifiée de catastrophiques. Le dysfonctionnement amène la poursuite du vol ou de l'atterrissage impossible, le crash de l'avion.

B : Dangereuses/difficiles - Dans ce niveau, l'occurrence d'une panne implique qu'elle est dangereuse. Exemples : blessures graves d'un ou

plusieurs occupants voire même la mort des occupants, problème physique/augmentation considérable de la charge de l'équipage, etc.

C : Majeures - Arrivé à ce niveau la panne devient majeure. La panne peut être une augmentation notable de la charge, de l'inconfort voire une blessure d'un (des) occupant(s), un dysfonctionnement des équipements vitaux de l'appareil.

D : Mineures - Á ce niveau la panne est qualifiée de mineure. Le dysfonctionnement peut amener un peu d'inconfort, une légère réduction des marges de sécurité, etc. mais rien qui ait un effet sur la sécurité du vol.

E : Sans effet - Ce niveau est le moins critique, si une panne survient elle n'a aucun effet sur la capacité opérationnelle et/ou aucun effet sur la charge de travail. Si le logiciel est considéré comme étant de niveau E, il est dit "hors cadre" de la norme. Le logiciel est, de ce fait, de niveau QM de la norme ISO 26262.

Ces différents niveaux sont appelés *DAL* pour ***D**evelopment **A**ssurance **L**evel*. On peut voir sur la figure 2.3 la relation qu'il y a entre les niveaux de criticité et leur fréquence.

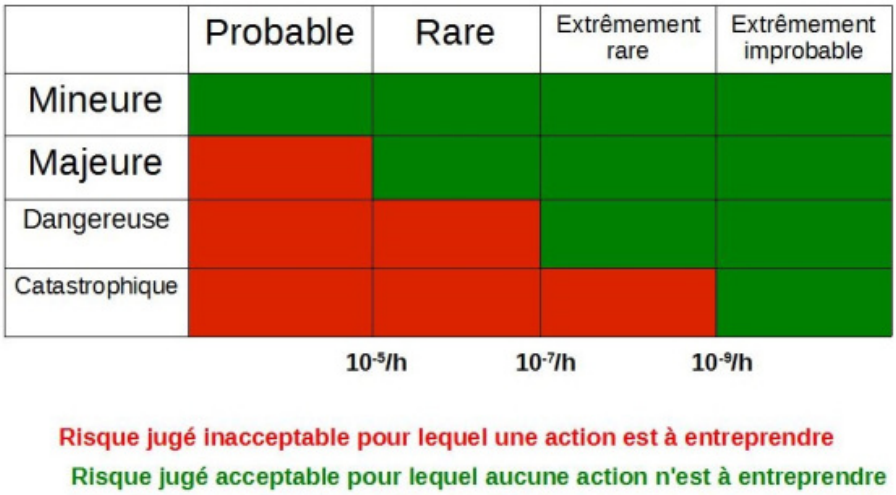


FIGURE 2.3 – Tableau des niveaux de criticité par rapport à leur occurrence possible avec une probabilité en nombre d'accident par heure de vol

Cette norme donne un ensemble de conditions de sécurité pour les systèmes critiques mais également des étapes de développement et des tests de logiciels.

Lorsqu'un niveau est attribué, il y a une série d'objectifs à atteindre[HAR16] :

E : Vu qu'à ce niveau une panne n'a aucun effet sur la sécurité du vol, il n'y a aucune contrainte particulière.

D :
- Le logiciel doit être documenté, les plans doivent être établis. Tout ce qui est spécifié doit être formellement vérifié. Tous les documents doivent être formellement vérifiés.
- Le logiciel doit être géré en configuration
- Un service Qualité indépendant doit assurer le respect de la norme

C : en plus des contraintes du niveau D :
- Des règles de développement doivent être fixées au préalable
- Les contraintes de traçabilité et de vérification s'appliquent également aux phases de conception et de codage du logiciel.
- Les exigences de bas niveau (ou exigences de conception) doivent être formellement vérifiées.
- Couverture des instructions (Statement Structural Coverage) ; Ces contraintes amènent souvent à devoir passer des tests unitaires.

B : en plus des contraintes du niveau C :
- Couverture de décision / Couverture de condition (Decision/Condition Structural Coverage)
- Les activités de développement et de vérification doivent être confiées à des équipes indépendantes.

A : en plus des contraintes du niveau B :
- Couverture de condition / Décision modifiée (Modified Condition Decision Structural Coverage, MC/DC)

2.3 Sûreté de fonctionnement

Cette section reprend la définition de la sûreté de fonctionnement, les propriétés de celle-ci, les méthodes pour développer un système sûr de fonctionnement, une classification des fautes élémentaires avec les définitions des différents types de faute. La section est basée sur les références [HH] [LAP], [GUI11].

2.3.1 Définition

Laprie a défini la sureté de fonctionnement en se basant sur la justification de la confiance, qui peut être définie comme une dépendance acceptée que cela soit implicite ou explicite.

La sûreté de fonctionnement d'un système est son aptitude à délivrer un service de confiance justifiée.

La dépendance qu'un système à d'un autre est en réalité la dépendance de la sûreté de fonctionnement de ce premier sur ce dernier.

Elle a une importance considérable dans l'étude des systèmes critiques puisqu'elle permet de maintenir le fonctionnement de ces systèmes. La définition de la sûreté de fonctionnement peut aussi se faire à travers l'arbre à la section suivante.

2.3.2 Arbre de la sureté de fonctionnement

Grâce à l'arbre visible à la figure 2.4, les différentes facettes de la sûreté de fonctionnement sont exposées.

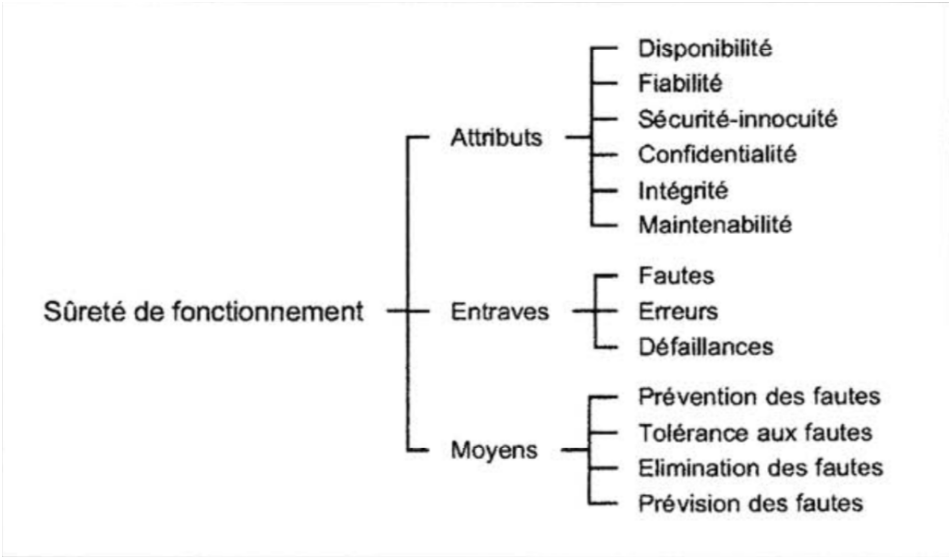


FIGURE 2.4 – Arbre de la sûreté de fonctionnement

Comme on peut le voir, les différents aspects sont les attributs, les moyens et les entraves.

2.3.2.1 Attributs

Les propriétés, citées précédemment, sont complémentaires bien que différentes et permettent de définir les attributs. Ceux-ci sont :

La disponibilité : défini par le fait d'être prêt à l'utilisation.

La fiabilité : continuité du service.

La sécurité-innocuité : inexistence de conséquences catastrophiques pour l'environnement.

La confidentialité : inexistence de divulgations non-autorisées de l'information.

L'intégrité : inexistence d'altérations non-appropriées de l'information.

La maintenabilité : défini par l'aptitude à réparer et à évoluer.

Si on reprend la définition de la sûreté de fonctionnement, le but est de fournir un service de confiance justifiée, c'est-à-dire, un service correct. Autrement dit, le système doit avoir le moins possible de défaillance de service, plus communément appelée défaillance. La définition d'une défaillance est

Un imprévu qui fait dévier le service délivré du service correct.

Sachant cela, il est possible de déduire que cette défaillance est dûe à un non-respect d'une spécification fonctionnelle **ou** à une spécification fonctionnelle qui ne décrit pas ou qui décrit mal la fonction que l'on souhaitait exprimer au départ.

De cette conclusion, la défaillance peut être définie comme

Une transition d'un service correct vers un service incorrect, qui ne remplit pas la fonction du système.

La défaillance est une **erreur** et une erreur a pour origine une **faute** comme on peut le voir en figure 2.5. Une erreur est définie comme

Une partie de l'état du système susceptible d'entraîner une défaillance.

FIGURE 2.5 – Chaîne représentant les relations entre faute, erreur et défaillance.

Pour revenir à la faute, elle est qualifiée de interne ou externe au système. Une faute interne est appelée vulnérabilité, elle permet à une faute externe d'avoir un impact sur le système et ainsi entraîner une défaillance. La-dite faute peut être de plusieurs natures :

Phase de création/conception : fautes de développement, fautes opérationnelles.

Frontières du système : fautes internes, fautes externes.

Cause phénoménologique : fautes naturelles, fautes dûes à l'homme.

Dimension : fautes matérielles, fautes logicielles.

Objectif : fautes malveillantes, fautes non-malveillantes.

Intention : fautes délibérées, fautes non-délibérées.

Capacité : fautes accidentelles, fautes d'incompétence.

Persistance : fautes permanentes, fautes temporaires.

Grâce à tout ce qui est défini auparavant, nous pouvons donner une définition complémentaire de la sûreté de fonctionnement qui est la suivante

Aptitude à éviter des défaillances du service plus fréquentes ou plus graves qu'acceptable.

Lorsque le système commence à avoir des défaillances trop fréquentes ou des défaillances qui dépassent un niveau dit acceptable, les défaillances en question marquent une défaillance de la sûreté de fonctionnement.

Lors du développement d'un système sûr de fonctionnement, il est possible d'utiliser plusieurs moyens afin d'éviter la source des défaillances, les fautes, et donc leur occurrence. Ces différents moyens sont :

Prévention des fautes : détermine le moyen d'empêcher l'occurrence d'une ou plusieurs fautes ou l'introduction de celles-ci.

Tolérances des fautes : explicite la manière pour que le système remplisse ses fonctions malgré la présence de fautes.

L'élimination des fautes : permet de réduire le nombre de fautes ainsi que leur niveau de criticité.

La prévision des fautes : met en évidence l'ensemble des méthodes afin d'estimer la présence, la conséquence et la création des fautes.

Toutes les notions définies ici sont résumées dans l'arbre présenté en figure 2.4.

3 Système interactif

Dans cette section, nous donnons la définition d'un système interactif selon les références suivantes [JC], [BL97], [Tor], [FJ14], [CNS+95].

Selon les auteurs du chapitre [JC], la définition d'un système interactif se résume à

Un système numérique qui est en interaction avec l'humain.

Comme on peut le voir à la figure 2.6 un tel système est composé de deux parties :

Noyau fonctionnel : ensemble de services dépendants d'un domaine d'application qui sert au traitement et au stockage de l'information.

Interface Homme-Machine : ensemble de fonctions logicielless et d'éléments matériels, servant d'intermédiaires entre le noyau fonctionnel cité ci-dessus et l'humain, qui sont mis en oeuvre lorsqu'il y a capture des entrées de l'utilisateur ainsi que lors de la restitution des sorties du système.

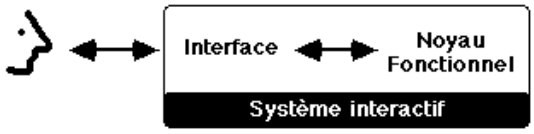


FIGURE 2.6 – Composition d'un système interactif

Les outils pour interagir avec cette interface, tels que le clavier et la souris, sont appelés ressources d'interaction ou modalités d'entrée (figure 2.7). Ces éléments peuvent varier selon l'interface. Dans notre cas, la ressource qui est utilisée pour interagir avec notre interface est le point de contact de notre main. Celui-ci apparaît lorsqu'un doigt de la main entre en contact avec notre interface tactile. L'interface possède une modalité de sortie ou système de rendu qui est l'écran tactile (figure 2.7).

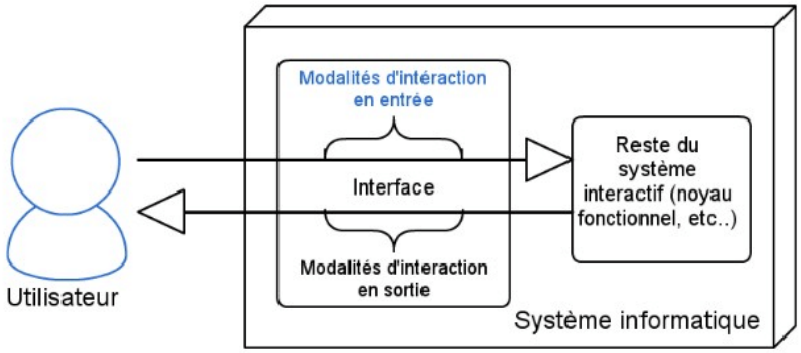


FIGURE 2.7 – Système interactif

Selon [Tor], le système interactif peut être traduit comme étant

Une application informatique qui considère les entrées des utilisateurs, donc les informations qui en découlent, et la représentation externe qui est produite par le système.

Selon l'angle que l'on choisit, il est possible de modéliser le système interactif de différentes façons. Ces aspects sont les suivants : le domaine d'application, l'architecture logicielle, les aspects ergonomiques et cognitifs.

Un système interactif est dit multimodale lorsque l'utilisateur a la possibilité d'avoir plusieurs moyens, plusieurs modalités d'interaction avec le système que celles-ci soit en entrée ou en sortie. Ces modalités peuvent être utilisées en parallèle ou non.

Une modalité est un moyen pour un utilisateur d'interagir afin d'atteindre un objectif.

Les outils qui seront utilisés dans ce travail sont focalisés sur la notation des descriptions de tâches. Tous les détails par rapport à ces outils seront rapportés dans la section 5.

4 Concept de similarité

Pour introduire le concept de similarité, la section débute par une sous-section expliquant le concept vu de manière générale et s'ensuit une autre sous-section avec le concept détaillé de manière plus formelle.

4.1 Description générale

La définition qui suit a été tirée du dictionnaire Larousse en ligne et va nous permettre d'exposer le principe de similarité.

"Se dit de choses qui peuvent, d'une certaine façon, être assimilées les unes aux autres." [2]

La plupart des personnes ont l'habitude, dans leur quotidien, d'utiliser le concept de similarité. En effet, ce mot est régulièrement utilisé de plusieurs manières différentes mais généralement il est utilisé de façon familière.

"Oh, regarde ce téléphone c'est le même que l'autre là-bas.
Tu as raison ils sont assez similaires vus comme ça. "

Prenons cet exemple.

Nous sommes dans une situation complètement informelle.

Les deux téléphones sont pris comme objets de comparaison.
La première personne montre le téléphone qu'elle est en train de regarder et le téléphone qu'elle a vu précédemment. Celle-ci fait la comparaison et inconsciemment les définit de similaires. La personne considère simplement que les deux téléphones se ressemblent esthétiquement et donne de manière informelle une définition de ce qu'est pour elle la similarité.
Mais peut-on vraiment attester que cela est suffisant afin de définir les deux objets de similaires ?

4.2 Description formelle

Introduisons le concept de similarité de manière plus formelle. Cette partie a été basée sur le papier de Cicekli sur les similarités et les différences [Ily], ainsi que sur la thèse de Sébastien Sorlin concernant la similarité de graphe [Sé06].

La similarité est utilisé dans de nombreux domaines comme la comparaison de graphes ou encore le machine learning.

Dans le travail de [Sé06], Sorlin expose le fait que les arbres sont régulièrement utilisés en tant que représentation de documents. Ils permettent de représenter les documents qui sont décomposables dont les sous-documents peuvent également être décomposables.
Un arbre possède une granularité et celle-ci dépend de la profondeur de l'arbre. Dès lors, il est possible de dire qu'un document (autrement dit, un noeud de l'arbre) est similaire à un autre si leurs décompositions en sous-documents sont similaires.

2. http ://www.larousse.fr/dictionnaires/francais/similaire/72788

L'article [Ily] montre la similarité entre deux phrases comme définit par Cicekli.
Voyons sa définition de similarité. Pour cela, il est nécessaire de définir quelques termes auparavant :

Alphabet A	C'est un ensemble non-vide de terminaux
Langage L	Sur un alphabet A est un ensemble non-vide fini de string non-vide avec une longueur finie.
Chaine de caractères	Une chaine de caractères sur A est un membre de A^*. Si c'est un membre de L, la chaine de caractères est appelée phrase de L.
Similarité	Une similarité entre deux chaines de caractères, où ces deux chaines de caractères sont des phrases de L, est une phrase non-vide β tel que $$\alpha_1 = \alpha_{1,1}\,\beta\alpha_{1,2} \quad (2.1)$$ et $$\alpha_1 = \alpha_{2,1}\,\beta\alpha_{2,2} \quad (2.2)$$. La similarité représente une partie similaire entre deux phrases.

4.3 Les systèmes de recommandations

Les systèmes de recommandation vont fonctionner par similarité.

4.3.1 Netflix

Si on prend, par exemple Netflix, ce genre de site propose un système de recommandation qui fonctionne comme suit : Un utilisateur lambda regarde un ensemble de films. Sur base des films regardés, Netflix proposera un autre ensemble de films qu'il considèrera comme intéressant pour cet utilisateur lambda. Oui mais comment un tel système de recommandation fonctionne ?

Reprenons notre utilisateur lambda, celui-ci regarde une série de films ou simplement les épisodes d'une série. Á côté de ça, nous avons d'autres utilisateurs. Ceux-ci ont regardés certains films que l'utilisateur lambda a regardé. Ensuite, ces utilisateurs ont regardés d'autres films ou séries que l'utilisateur lambda. Ces "autres" films/séries sont proposés à l'utilisateur lambda. Ceci puisque l'utilisateur lambda a regardé les mêmes films que les autres.

De manière plus formelle, définissons :
— un utilisateur U
— des utilisateurs $U_1, U_2, ..., U_n$
— un film regardé par un utilisateur U : f
— une série de films regardés par un utilisateur U : $\{f_1, f_2, ..., f_n\}$
Un utilisateur U_1 regarde un ensemble de films $\{f_1, f_2, f_3, f_4, f_5\}$. Un autre utilisateur U_2 regarde les films $\{f_2, f_4\}$.

Comme l'utilisateur U_1 a regardé $\{f_1, f_2, f_3, f_4, f_5\}$ et l'utilisateur U_2 a regardé $\{f_2, f_4\}$. L'utilisateur U_2 va recevoir comme recommandation tout ce qui se trouve dans l'ensemble U_1 mauve, en dehors de l'intersection rouge :

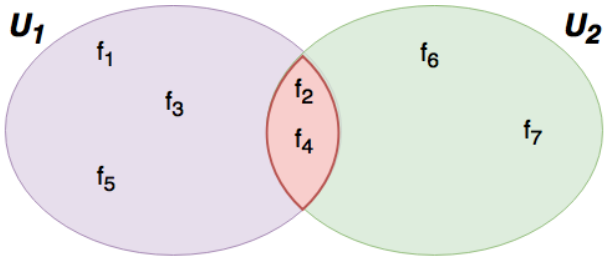


FIGURE 2.8 – Intersection des films regardés par les utilisateurs

Donc, U_2 aura comme suggestion : $\{f_1, f_3, f_5\}$ et U_1 lui aura : $\{f_6, f_7\}$. C'est ce qu'on appelle le filtre collaboratif[RMWZ14].

4.4 L'heuristique de similarité

En rapport avec la section 4.1, il y a l'heuristique de similarité [RGC11].

Les "décideurs", qui utilisent l'heuristique de similarité, vont juger la probabilité qu'une instance est membre d'une catégorie plutôt qu'une autre par le degré de similarité des autres dans cette catégorie. Le principe de l'heuristique de similarité est illustré par un exemple à la sous-section **Exemple** ci-dessous.

Selon les temps de réponse de cette étude[RGC11], on conclut que c'est comme ça que les choix sont faits. Toujours selon cette étude, cette heuristique de similarité tend à être une règle de choix fiable et précise.

L'étude explique que cette façon de faire des choix grâce à la similarité est quelque chose avec laquelle nous sommes né. C'est une façon de faire que nous utilisons dès que nous sommes petit. Donc, il est facile pour nous d'utiliser ce principe de l'heuristique de similarité.

4.4.1 Exemple

Prenons comme exemple les personnes qui achètent des voitures de la marque Mercedes [3].

Ces personnes trouvent les voitures Mercedes de bonne qualité, fiable, etc. Donc, lorsque ces personnes vont voir d'autres produits avec cette même marque sur l'étiquette comme de l'huile ou autres, ils vont penser que celles-ci sont de qualité également.
Cette réflexion est faite car leur voiture de la même marque est de qualité.

De ce fait, ils vont avoir tendance à rester dans ce qu'ils connaissent déjà comme marque/qualité.

3. Une autre marque aurait très bien pu être prise pour l'exemple.

4.5 Dans notre cas

Comme expliqué précédemment à la section 2.3, en sûreté de fonctionnement, l'une des préoccupation principale, à l'origine, est le fait de fournir un service de confiance justifiée[LCG89]. En réalité, la préoccupation est d'éviter au maximum des défaillances du système trop fréquentes ou des défaillances aux conséquences graves.

Dans notre domaine, qui est pour rappel l'aviation, un des moyen pour éviter les défaillances est, comme cité en section 2.1.1.1, la redondance. Cette redondance doit permettre d'utiliser une fonction du système via plusieurs modalités. Et ces modalités doivent faire preuve d'un degré de similarité élevé. La similarité doit apparaitre au maximum tant au niveau visuel qu'au niveau interaction.

L'étude de la similarité est très importante [NPHDP] à ce niveau tant au niveau entrée qu'au niveau sortie même si les types d'écrans ou les types de dispositifs de capture d'entrée sont différents. La similarité est très spéciale dans notre cas car elle n'a de sens que lorsque deux systèmes interactifs sont considérés (ou deux versions différentes d'un même système interactif).

Cette similarité est étudiée plus loin dans ce mémoire.

5 Modèle de tâches

Dans cette section, plusieurs questions sont abordées : qu'est-ce qu'un modèle de tâches, quels sont les outils possibles pour faire ce type de modélisation, pourquoi avoir choisi cette modélisation plutôt qu'une autre, pourquoi avoir choisi un outil en particulier, etc.

Cette section ainsi que ces sous-sections se réfèrent aux papiers suivants :[Bla11], [GGSL92], [Bal94], [SGGC08].

5.1 Définitions

Dans un domaine d'application complexe tel que celui de l'aviation, il est très important de bien comprendre les activités liées aux métiers. Ce fait de comprendre les activités liées aux métiers consiste, en fait, en une analyse de tâches que nous définirons plus loin.

Avant de définir ce qu'est un modèle de tâches, il est nécessaire de définir quelques notions auparavant.

Commençons par le concept de tâche. Selon [Bla11],

Une tâche consiste en un but (état souhaité) et une procédure pour atteindre le but.

Une tâche est un ensemble de traitements et que chacun de ces traitements permet de décrire un moyen d'atteindre le but fixé de la tâche[GGSL92]. Un traitement est défini par la description de [GGSL92] :

Ses entrées : les valeurs admises pour lancer l'exécution de la tâche selon ce traitement particulier.

Ses sorties : les valeurs attendues comme résultats.

Du traitement lui-même : décomposition de la tâche en sous-tâches.

Comme cité ci-dessus, une tâche est décomposée en sous-tâches et la décomposition s'applique également aux sous-tâches de cette dernière. La décomposition est qualifiée, de ce fait, de récursive. La décomposition s'arrête par une tâche dite terminale, associée à une procédure.

Une procédure est définie comme

Un ensemble de sous-tâches liées par des relations de composition et des relations temporelles.

Un ensemble de procédures est défini lorsqu'on se trouve sur le plus bas niveau de la décomposition d'une tâche. Ces procédures sont définies comme des tâches terminales. L'exécution d'une tâche équivaut à l'exécution de l'ensemble de ses tâches terminales dans l'ordre donné par la décomposition.

Dans un modèle de tâches, il y a plusieurs types de tâches :
— les tâches concrètes,
— les tâches concrètes terminales,
— les tâches abstraites.

Une tâche **concrète** est une tâche qui n'a qu'un seul traitement afin d'atteindre un but concret. Une tâche **concrète terminale** est une tâche qui ne peut être décomposée et qui, par définition, décrit un appel à une procédure. Une tâche **abstraite** qui regroupe deux sortes de tâches :

Des tâches abstraites qui ont un caractère peu général (peuvent décrire le même but concret ou plusieurs buts concrets différents),

Des tâches concrètes qui ont le même but concret.

Lorsqu'une tâche abstraite est décomposée en d'autres tâches abstraites, celle-ci a un contexte plus général que celles qui la composent.

[Bla11] aborde aussi la notion de tâche élémentaire. Une tâche élémentaire est

Une tâche qui peut être décomposée en diverses actions physiques.

De par cette définition, une action physique est définie comme

Une opération sur un dispositif d'entrée ou de sortie qui provoque un changement d'état de ce même dispositif.

Enfin, nous pouvons donner la définition de modèle de tâche selon [Bla11],

Un modèle de tâches est une structure en arbre qui possède des noeuds représentant les buts et des sous-arbres qui sont les procédures pour atteindre les dits buts.

D'après [Bal94], le modèle de tâche sert lors de la conception mais sert également à faire de la documentation pour l'utilisateur. Mais il ne sert pas seulement à ça, il est également très utile pour valider des IHM[SGGC08].

5.2 Choix du modèle de tâches

Comme expliqué à la section précédente 5.1, il est très important de faire l'analyse de tâches lorsqu'on étudie les activités d'un métier aussi complexe que celui de pilote. De par sa dénomination, il est clair que l'analyse de tâche consiste en la description de tâches qui sont effectuées pour accomplir un but, un travail.

Pour acquérir les différentes informations liées à l'activité ciblée, il y a plusieurs façons de procéder :

— l'introspection,
— les questionnaires,
— les interviews ex-situ [4],
— les interviews in-situ [5],
— l'observation,
— ...

Quelle que soit la méthode choisie, il faut bien garder à l'esprit qu'il faut différencier ce qu'il se passe réellement de ce qu'il se passe en théorie, selon les règles.

4. hors site, pas en situation d'activité
5. sur place, en situation d'activité

Par exemple, lors d'une interview avec un caissier [6] celui-ci vous dira qu'ils ne peuvent pas s'échanger d'argent entre caissiers simplement parce que c'est le règlement. En effet, ils ont peur que ce qu'ils font vraiment soit rapporté au patron donc ils disent ce qu'ils sont censés faire. Or, dans la réalité, si vous allez les observer sur le terrain, vous verrez qu'en fait lorsqu'il manque un peu de monnaie à une caisse, les caissiers n'hésitent pas à s'échanger entre eux de la monnaie pour ne pas devoir appeler le responsable et perdre du temps avec cela.

Pour transmettre ou communiquer ces informations, il faut que celles-ci soient capturées. Évidemment il existe plusieurs outils afin de permettre ceci :

— les scénarios de travail,
— les uses cases d'UML,
— les modèles de tâches.

Il est clair que le choix qui a été fait ici est celui du modèle de tâches. Le choix du modèle de tâches est dû au fait que celui-ci permet de **formaliser** non seulement les buts de l'utilisateur mais également la manière dont il les atteint et de quelles informations il a besoin pour les atteindre. En effet, ce qui est souhaité ici c'est d'avoir absolument toutes les étapes quelles qu'elles soient comme les intentions, etc. Tandis que le scénario de travail permet de décrire cela de manière moins formelle et le use case ne donne pas les informations telles que, par exemple, les connaissances utilisées lors de l'action. De plus, vu notre domaine d'application, nous avions besoin de garantie de sécurité. Or, le modèle de tâches permet de décrire de manière formelle l'ensemble des actions de l'utilisateur et, de ce fait, nous apporter la garantie dont nous avions besoin.

5.3 Outils

Comme pour beaucoup de technologies, de méthodes, il existe plusieurs outils possibles afin de modéliser les tâches. Le but de cette section est d'en énumérer une partie non-exhaustive. Cette section se base principalement sur le travail [FJ14], [CSS08], [Pat04], [GCS], [RCP14].

5.3.1 K-MADe

K-MAD[FJ14] [GCS] pour **K**ernel of **M**odel for **A**ctivity **D**escription est une notation qui possède des héritages de notations déjà existantes dont CTT décrit à la section 5.3.2.

C'est un formalisme qui permet de décrire des activités humaines via des arbres de tâches avec une sémantique formelle [CSS08]. Dans les concepts de ce formalisme, on retrouve : les tâches, les objets et les expressions, les évènements, les utilisateurs.

Dans le formalisme, quatre types de tâches sont définies : abstrait, utilisateur, système et interactif.

Les tâches sont caractérisées par plusieurs choses :

— numéro,
— nom,
— but,
— exécutant,
— ...

6. ou une caissière, l'exemple masculin est pris par facilité

Tout comme le CTT, la décomposition d'une tâche donne l'ordre d'exécution des sous-tâches. Dans la notation, il est également possible d'assigner à une tâche une expression qui est considérée comme condition (pré, post, etc). Ces conditions sont exprimées à l'aide d'objets. Ceux-ci peuvent être :

Abstrait : c'est-à-dire composés de caractéristiques manipulés par l'utilisateur,

Concret : c'est-à-dire que ce sont des instances des objets abstraits.

Un outil graphique a été développé pour pouvoir modéliser des tâches avec la notation K-MAD, celui-ci est appelé K-MADe.

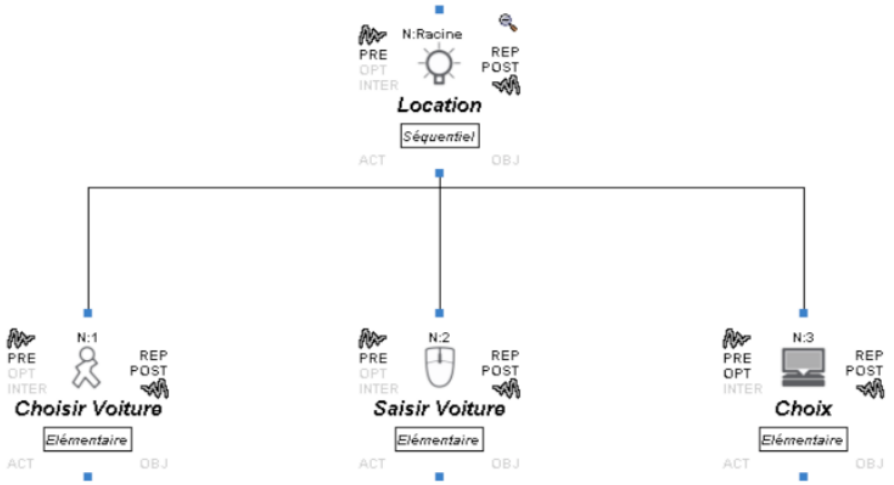


FIGURE 2.9 – **K**ernel of **M**odel for **A**ctivity **D**escription environment

5.3.2 CTTe

CTT[FJ14] est une notation de modélisation de tâches tandis que CTTe est son environnement d'édition et d'analyse de modèle de tâches. Cet outil est utile afin de soutenir la conception d'applications interactives en modélisant les activités de l'humain.

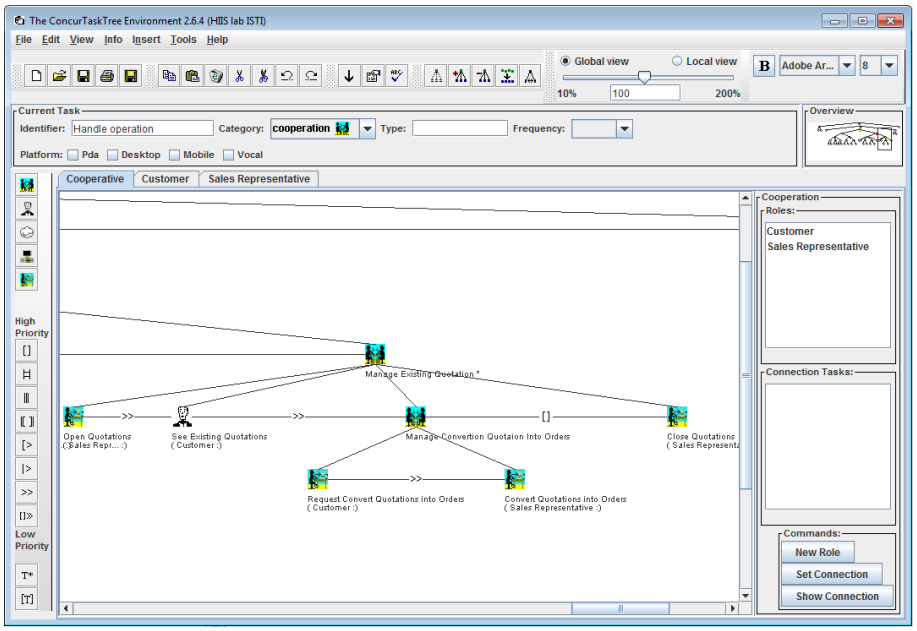


FIGURE 2.10 – ConcourTaskTree Environment

Comme on peut le voir à la figure 2.10, la notation décrit quatre types de tâches :

FIGURE 2.11 – Tâche abstraite

FIGURE 2.12 – Tâche d'interaction

FIGURE 2.13 – Tâche utilisateur

FIGURE 2.14 – Tâche système

Mais cette notation définit également plusieurs opérateurs qui sont les suivants [Pat04]

Opérateur	Description
T1 » T2	Spécifie le fait que la tâche T2 ne peut être effectuée avant la tâche T1
T1 [•] T2	Spécifie que si la tâche T1 est exécutée la tâche T2 n'est plus disponible
T1 [•]» T2	Spécifie que la tâche T2 ne peut être effectuée avant que la tâche T1 soit terminée ET que la sortie de l'exécution de T1 sera utilisé en entrée pour T2
T1 ||| T2	Spécifie que les tâches T1 et T2 peuvent s'effectuer dans n'importe quel ordre ou au même moment, incluant la possibilité qu'une tâche commence avant qu'une autre ne soit terminée
T1 |[•]| T2	Les tâches peuvent échanger de l'information pendant qu'elles sont exécutées en concurrence.
T1 |=| T2	Les tâches peuvent être exécutées dans un ordre quelconque mais une fois qu'une tâche a commencé elle doit se terminer pour que l'autre s'exécute
T1 [> T2	La tâche T1 est interrompue définitivement lorsque la tâche T2 commence
T1 |> T2	La tâche T1 peut être interrompue par T2 et quand T2 est terminée T1 peut être réactivé selon l'état stocké

La particularité de CTT est qu'il exprime la composition entre tâches descendantes d'une tâche parent plutôt que de l'exprimer sur le plan de la tâche parent [FJ14].

De par ses notations et ses possibilités, cet outil ne peut être retenu comme choix d'outil de modélisation. Celui-ci n'est pas assez riche pour notre domaine d'application et pour les informations souhaitées.

5.3.3 Hamsters

Comme pour les outils présentés précédemment, Hamsters[FJ14] [RCP14] [IRI] est un outil de modélisation de tâches utilisateurs et il emprunte également des concepts dans des formalismes existants.

Cependant, Hamsters a une granularité plus fine au niveau des types de tâches qu'il propose. En effet, les tâches interactives ont été raffinées ainsi que les tâches humaines et les tâches collaboratives.

Hamsters est en constante évolution afin de permettre à ses utilisateurs de décrire des tâches toujours plus finement.

Cet outil permet de gérer un très grand nombre de tâches ainsi que des tâches complètes. Effectivement, l'outil permet de créer des composants, des sous-modèles, etc. Ce qui est innovant par rapport aux outils déjà cités.

Hamsters promouvoit son utilisation lors de toute la durée de la conception d'un système interactif allant de l'analyse des besoins jusqu'à la spécification

d'un système en passant par l'analyse des activités de l'utilisateur.

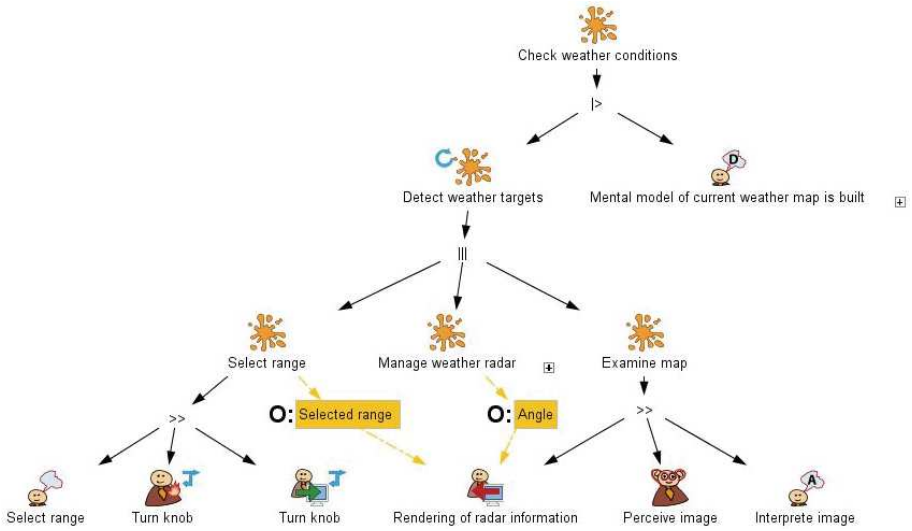


FIGURE 2.15 – Modèle de tâche HAMSTERS pour l'objectif "Check weather condition"

Pour cet outil, la notation est étudiée plus en profondeur que pour les précédents outils puisque cela nous servira dans la suite de cet ouvrage. Dans le tableau qui suit 2.16, les tâches sont citées de manière non-exhaustive :

	Abstract	Input	Output	IO	Processing	Group
Abstract		Not applicable	Not applicable	Not applicable	Not applicable	
User Individual						
User Cooperative						
Interactive Individual					Not applicable	
Interactive Cooperative					Not applicable	
System						

FIGURE 2.16 – Type de tâches HAMSTERS

Les tâches sont classées selon 6 catégories :
— tâches abstraites,
— tâches d'entrées,
— tâches de sorties,
— tâches d'entrées et de sorties,

— tâches de "process",
— tâches de groupe.
Dans les tâches de "process" se trouve la tâche représentée à la figure 2.17.

FIGURE 2.17 – Tâche cognitive individuelle

Ce type de tâche peut être redivisé en deux catégories : l'analyse et la décision. Celles-ci peuvent également être redivisé en catégorie individuelle et catégorie coopérative comme on peut le voir dans les figures ci-dessous 2.18, 2.19, 2.20, 2.21.

FIGURE 2.18 – Tâche d'analyse individuelle

FIGURE 2.19 – Tâche d'analyse coopérative

FIGURE 2.20 – Tâche de décision individuelle

FIGURE 2.21 – Tâche de décision coopérative

La liste montrée précédemment est dite non-exhaustive. Puisque, dans un premier temps, pour les besoins du projet un type de tâche a été ajoutée comme on peut le voir en figure 2.22. Hamsters peut ainsi évoluer en fournissant de nouveaux types de tâches.

FIGURE 2.22 – Tâche utilisateur d'attente

Dans un deuxième temps, d'autres tâches ne sont pas présentes dans la liste parmi ceux-ci on retrouve :

FIGURE 2.23 – Mouvement de tête vers le haut

FIGURE 2.24 – Mouvement de tête vers le bas

FIGURE 2.25 – Mouvement de tête vers la gauche

FIGURE 2.26 – Mouvement de tête vers la droite

FIGURE 2.27 – Mouvement de tête vers le haut

FIGURE 2.28 – Mouvement de tête vers le bas

FIGURE 2.29 – Mouvement de pression avec un doigt de la main gauche

FIGURE 2.30 – Mouvement de relâche du doigt de la main gauche

FIGURE 2.31 – Mouvement de pression avec la main gauche

FIGURE 2.32 – Mouvement de relâche de la main gauche

FIGURE 2.33 – Mouvement d'étendre le bras gauche

FIGURE 2.34 – Mouvement de ramener le bras gauche

FIGURE 2.35 – Mouvement désignant l'action de saisir avec deux doigts de la main gauche

Toutes les icônes précédentes sont également disponibles pour le bras droit et la main droite.

FIGURE 2.36 – Icône représentant une information manipulée

FIGURE 2.37 – Icône représentant une connaissance nécessaire à l'action

FIGURE 2.38 – Icône représentant un périphérique d'entrées/sorties manipulé

FIGURE 2.39 – Icône désignant un objet manipulé

FIGURE 2.40 – Icône désignant l'action de voir

FIGURE 2.41 – Icône désignant l'action d'entendre

FIGURE 2.42 – Composant permettant de modulariser le modèle de tâches

FIGURE 2.43 – Icône permettant de faire un sous-modèle

Toutes les icônes de Hamsters ne sont pas explicitées dans ce travail. En revanche, celles qui sont nécessaires à la compréhension du travail le sont. Il y a trois options possibles à activer sur une tâche :

— rendre la tâche itérative (voir figure 2.44),
— rendre la tâche optionnelle (voir figure 2.45),
— rendre la tâche itérative ET optionnelle (voir figure 2.46).

FIGURE 2.44 – Tâche itérative

FIGURE 2.45 – Tâche optionnelle

FIGURE 2.46 – Tâche itérative et optionnelle

Pour les opérateurs, le site de l'IRIT [IRI]répertorie ceux-ci :

Opérateur	Description
T1 » T2	La tâche T2 est exécutée après la tâche T1
T1 [•] T2	La tâche T1 est exécutée ou la tâche T2 est exécutée
T1 ||| T2	Les tâches T1 et T2 sont exécutées au même moment
T1 |=| T2	La tâche T1 est exécutée puis la tâche T2 l'est OU la tâche T2 est exécutée puis la tâche T1 l'est
T1 [> T2	L'exécution de la tâche T2 interrompt l'exécution de la tâche T1
T1 |> T2	L'exécution de la tâche T2 interrompt l'exécution de la tâche T1 et l'exécution de la tâche T1 est reprise à la fin de celle de T2

5.4 Conclusion du chapitre

Maintenant que nous avons défini tous les concepts importants pour la suite du travail tels que le concept de système critique, de similarité ou encore de modèle de tâches et que nous nous situons par rapport à ce qui existe déjà, nous pouvons voir comment tous ces concepts sont mis en pratique afin de répondre à la problématique du travail.

Troisième partie

Contributions

Chapitre 3

Approche à base de modèles et de prototypes pour concevoir et évaluer des interactions similaires

Sommaire

Comme expliqué dans le chapitre d'introduction, nous nous intéressons au cockpit interactif. Nous avons un intérêt plus particulier pour l'étude des similarités ou dissimilarités qu'il y a entre différents types d'interfaces qui sont intégrables dans le cockpit. Après avoir posé tous les concepts dans la partie état de l'art, nous prenons une étude de cas afin d'analyser ces similarités pour les types d'interfaces étudiés à savoir l'interface physique, présente dans le cockpit, et l'interface tactile, qui va être imaginée et développée dans ce travail.

1 Etude de cas

Afin de limiter le périmètre du travail, nous avons décidé de prendre un cas particulier. Dans le cockpit, comme expliqué en section 1.1, il y a beaucoup de fonctionnalités complexes présentes.

Pour notre cas, nous allons nous limiter au panneau plafond et plus précisément à la partie du panneau concernant l'extinction des feux moteurs. La section est divisée en sous-sections comme suit :

Description informelle : comme l'indique son nom, cette sous-section a pour but de donner une description informelle du "fire button" ainsi que de la GoPro et de son intérêt ici,

Prototypes : ici, l'objectif est de donner un aperçu des prototypes imaginés, pour montrer un nouveau type d'interaction qui sera un maximum similaire, et d'en montrer l'évolution,

Description des tâches : cette sous-section explicite l'ensemble des tâches présentes dans le modèle de tâches, elle expliquera également pourquoi nous avons décidé de descendre a un tel niveau de granularité, ...

Modèles systèmes : cette sous-section aborde les modèles systèmes sans trop entrer dans le détail, cependant le pourquoi ces modèles systèmes ont été développés est présent,

Analyse de l'étude de cas : finalement, cette sous-section conclue sur les résultats obtenus de tout le travail présent précédemment dans la section.

1.1 Description informelle

1.1.1 Fire Button

Comme évoqué dans la description de la section générale, nous nous intéressons dans ce travail à un cas particulier qui est l'extinction des feux moteurs. Pour l'étude de ce cas et la description des tâches, il est important de bien comprendre comment fonctionne le bouton.

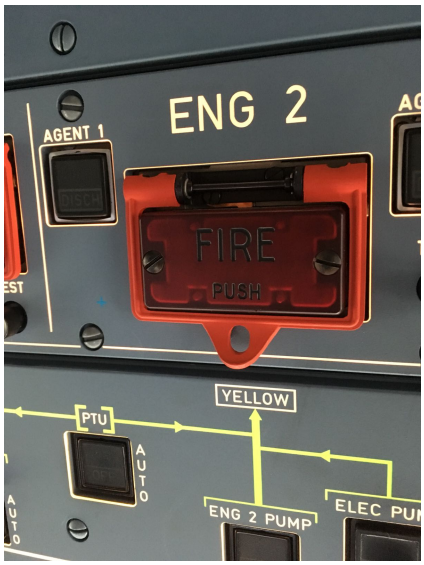


FIGURE 3.1 – Ensemble de boutons pour l'extinction d'un feu sur le moteur 2

Comme on peut le voir en figure 3.1, 3.2 et 3.3, l'extinction d'un (seul) feu moteur se fait via trois boutons. Chaque moteur dispose de cet ensemble de trois boutons. Les trois boutons sont :

— "Fire push" qui est notre "fire button",
— Agent1,
— Agent 2.

FIGURE 3.2 – Clapet fermé du même ensemble de boutons qu'en figure 3.1

Il est possible de l'observer en figure 3.2, le bouton "fire button" possède un certain relief ainsi qu'un clapet qui le protège. De ce fait, la protection et la sécurité de ce bouton sont doubles. Comme expliqué en 1.1.2, le clapet doit

être remonté et le bouton doit être pressé. Un détail qui n'est pas visible sur les illustrations est que le clapet s'il n'est pas remonté à son maximum redescend automatiquement.

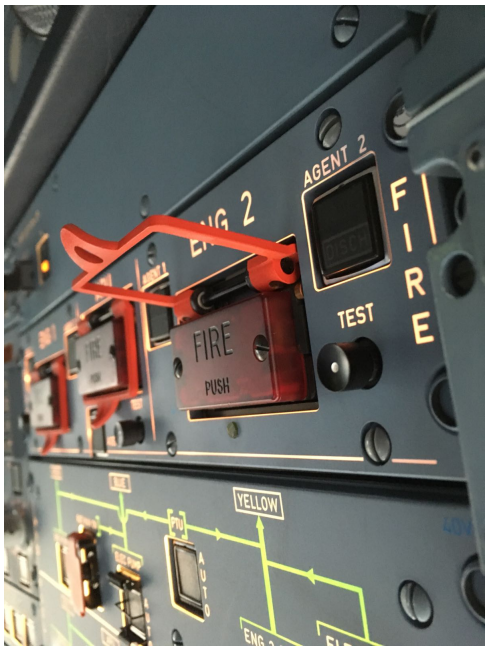


FIGURE 3.3 – Clapet ouvert du même ensemble de boutons qu'en figure 3.1

Lorsque le bouton est pressé, celui-ci s'allume en rouge et les boutons des agents sont en états "squib". Un agent peut être défini comme une sorte d'extincteur. L'état "squib" représente, de ce fait, l'état "prêt à être déchargé". Á partir de là, il est possible de décharger ces agents simplement en appuyant sur ceux-ci.

Les détails de la procédure sont expliqués ci-après à la section 1.1.2.

1.1.2 Procédure d'extinction d'un feu moteur

1. Une alarme sonore est déclenchée ainsi que l'alarme visuelle des boutons qui s'allument en rouge ET qui clignotent lorsqu'un feu est détecté. Ces deux boutons sont appelés "Master Warning" et "Master caution". Ces deux alarmes ne s'arrêtent qu'une fois l'un des deux boutons appuyés.
2. Une fois le feu détecté, on a un bouton du panneau plafond, correspondant au moteur dans lequel le feu est détecté, qui s'allume en rouge.
3. Le pilote/copilote déclare le feu à son équipier et au contrôle aérien.
4. Le pilote/copilote déclare le déclenchement de la procédure.
5. Le pilote/copilote éteint le moteur concerné.
6. Le système commence un compte à rebours.
7. A la fin de celui-ci, le pilote/copilote relève la protection du bouton moteur allumé.

8. Le pilote/copilote appuie sur le bouton pour armer l'agent (ou les agents), le bouton sort. Les deux boutons des agents s'allument.
9. Le pilote/copilote vérifie que les agents sont armés.
10. Le pilote/copilote appuie sur le bouton de l'agent pour le décharger.
11. L'écran du milieu (ATC Communication) est notifié.
12. SI le feu persiste après X secondes alors le second agent doit être déclenché de la même manière.

1.2 Description de tâches initiale

Une fois que la procédure identifiée, il faut formaliser tout cela et cela est fait grâce au modèle de tâches. Comme cité plus haut en section 5.3.3, le choix de l'outil s'est posé sur Hamsters. Les explications du modèle de tâches partent de l'hypothèse que c'est le pilote qui réalise la tâche. Évidemment le co-pilote peut très bien réaliser les actions qui figurent dans ce modèle [1].

A la figure 3.5 se trouve un modèle de tâche qui a simplement été réduit aux tâches qui sont "racines", qui se trouvent à la base de l'arbre.

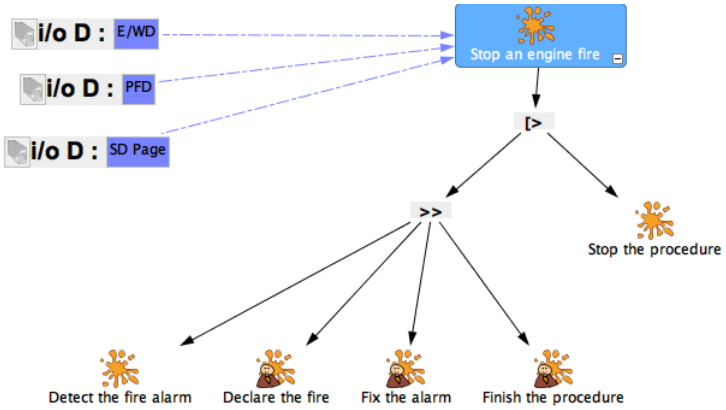


FIGURE 3.4 – Modèle de base réduit aux tâches racines

Ce modèle montre que la tâche principale qui est d'éteindre un feu moteur a été décomposée en sous-tâches. L'extinction d'un feu moteur c'est une séquence de tâches qui sont les suivantes : détecter l'alarme de feu, déclarer le feu, fixer l'alarme, finir la procédure **OU** bien le pilote/copilote décide d'arrêter la procédure à tout moment.

Avec la syntaxe de la modélisation donnée à la section 5.3.3, on voit qu'il s'agit de quatre tâches abstraites. Celles-ci sont donc elles-même décomposées en une séquence de sous-tâches. La première de ces sous-tâches est la suivante :

1. à l'exception d'annoncer le feu au co-pilote qui deviendra annoncer le feu au pilote

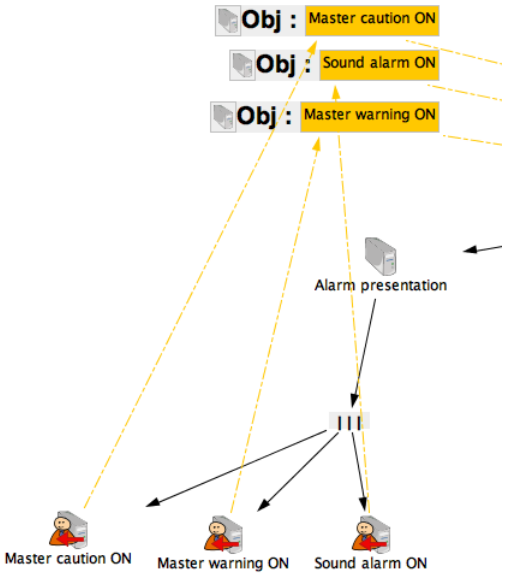


FIGURE 3.5 – Présentation de l'alarme par le système

Comme on peut le voir, c'est une tâche abstraite du système qui consiste en 3 actions :

Master caution ON : allumer le master caution,

Master warning ON : allumer le master warning,

Sound alarm ON : faire sonner l'alarme,

Ces actions ont pour résultats des objets qui vont être manipulés pas la suite. Ces objets ont le même nom que les tâches citées ci-dessus.

La seconde tâche est la perception de l'alarme par l'utilisateur (voir figure 3.6) qui est, dans ce cas, le pilote.

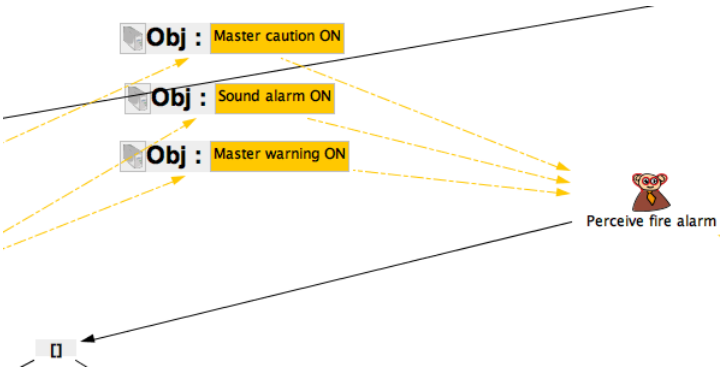


FIGURE 3.6 – Perception de l'alarme par le pilote

Cette perception se fait grâce aux entrées qui sont les 3 objets évoqués précédemment, c'est-à-dire : le master warning en état ON, le master caution en état ON, l'alarme sonore en état ON. À partir de ces objets, il peut clairement

percevoir qu'il y a une alarme de feu moteur. Seulement, il y a plusieurs façons différentes d'arriver à la perception de l'alarme qui sont représentés par l'opérateur de choix :

- figure 3.7 : le pilote voit le master warning allumé, l'alarme sur l'écran de notification "Engine SD page" et entend l'alarme sonore,
- figure 3.8 partie A : le pilote entend uniquement l'alarme sonore,
- figure 3.8 partie B : le pilote voit le master warning allumé et le master caution allumé,
- figure 3.8 partie C : le pilote voit l'alarme sur l'écran de notification "Engine SD page",
- figure 3.8 partie D : le pilote voit le master warning allumé.

Les tâches de perception visuelles et auditives reçoivent un périphérique de sortie en entrée qui est nécessaire à la perception. Toutes les tâches de perception visuelles produisent une information, cette information étant ce qu'aperçoit le pilote.

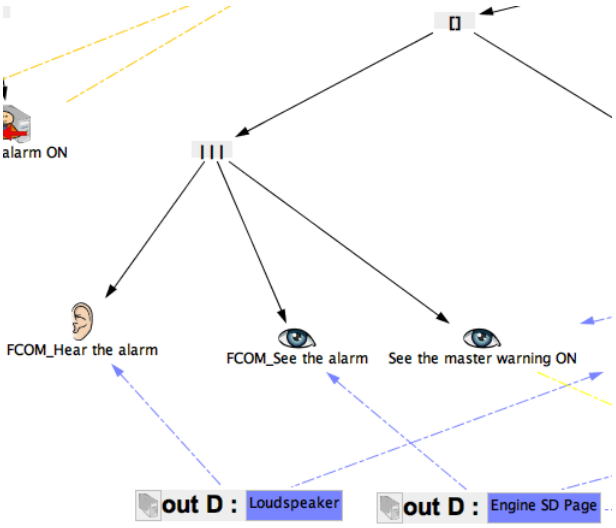


FIGURE 3.7 – Première possibilité de perception de l'alarme par le pilote

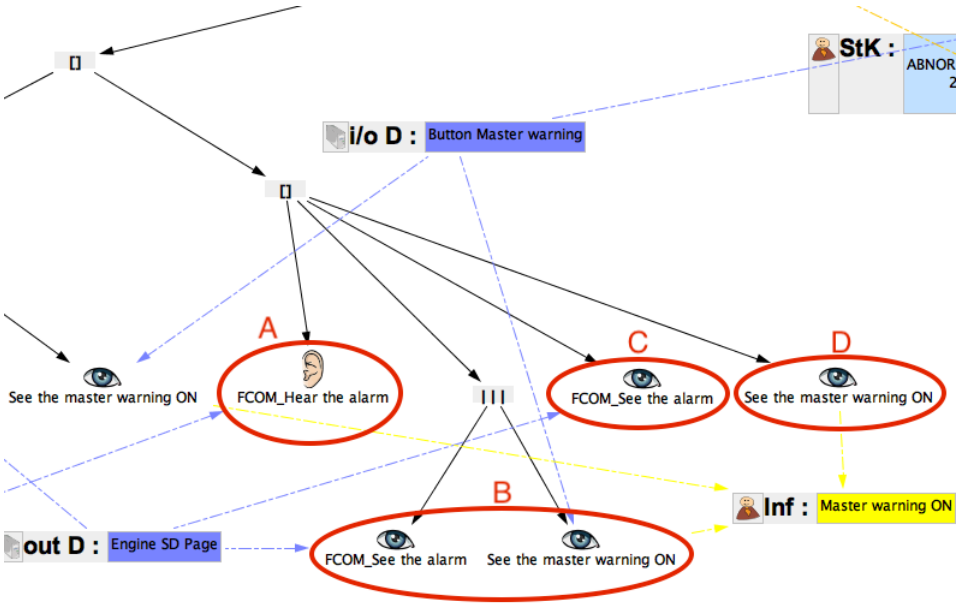


FIGURE 3.8 – Autres possibilités de perception de l'alarme par le pilote

Revenons à la séquence pour la perception de l'alarme par le pilote. Celle-ci se poursuit comme on peut le voir sur la figure 3.10.

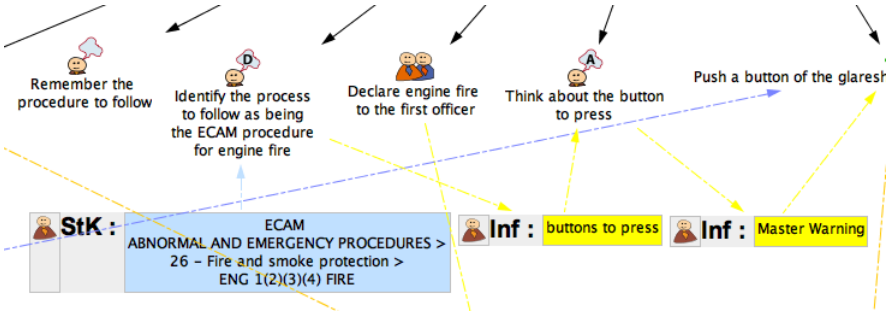


FIGURE 3.9 – Autres tâches de la séquence pour la perception de l'alarme

Le pilote doit se souvenir de la procédure à appliquer. Ensuite, il identifie la procédure à suivre comme étant celle présente dans l'ECAM à la section "ABNORMAL AND EMERGENCY PROCEDURES" ; celle-ci s'appelle ENG 1(2)(3)(4) FIRE et elle se trouve dans la sous-section "26 - Fire and smoke protection". Par après, il déclare le feu moteur à son co-pilote, à ce moment là le résultat en sortie est une information de feu potentiel. Il réfléchit au bouton sur lequel il doit appuyer.

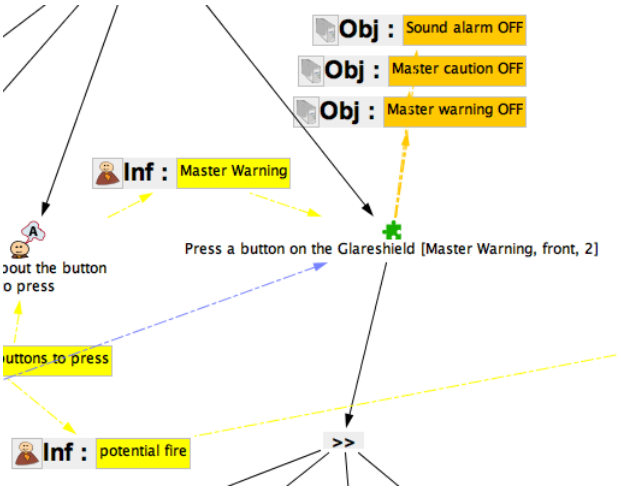


FIGURE 3.10 – Autres tâches de la séquence pour la perception de l'alarme

La dernière tâche de la séquence est un composant qui rend modulaire la tâche d'appui sur un bouton du "Glareshield", autrement dit le panneau en face du pilote et co-pilote. Ce composant pourra être réutilisé dans tout le modèle de tâche.

Pour une question de facilité et de compréhension toutes les tâches du composant sont modélisées en dessous du composant. En utilisation normale, le composant doit être instancié avec les paramètres souhaités.

Ce composant est constitué d'une séquence d'un composant et de tâches comme on peut le voir en figure 3.11.

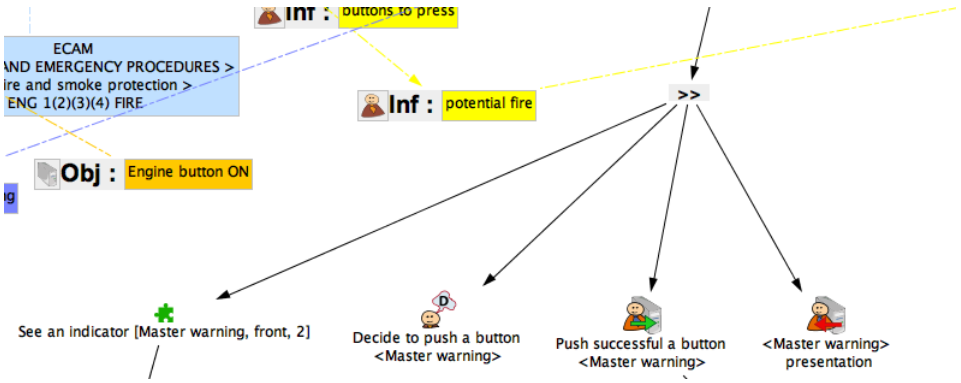


FIGURE 3.11 – Composant de la tâche d'appui sur un bouton du Glareshield qui prend en paramètre le bouton et sa position

La séquence commence par le composant en figure 3.11. Celui-ci modularise la tâche "Voir un indicateur". Il est régulièrement utilisé dans le modèle. Celui est également composé d'une séquence de tâches, comme on peut le voir en figure 3.12.

Cette séquence est divisée en un choix entre bouger la tête vers le haut (le bas, la gauche ou la droite) et décider ne rien faire. La tâche pour le choix du

mouvement de la tête est choisie en fonction du paramètre donné dans l'instance du composant qui, dans l'exemple, est "front". Une fois que ce choix est fait, le système effectue le rendu de l'indicateur en paramètre et le pilote étant dans la bonne position perçoit l'indicateur qui, ici, est le master warning.

Une fois que le pilote perçoit l'indicateur, il décide d'appuyer sur celui-ci (voir figure 3.13), il appuie avec succès et l'indicateur, ici le master warning, a un nouveau rendu. L'appui se fait de la manière suivante : il étend le bras, il appuie sur le bouton master warning avec son doigt et il ramène son bras.

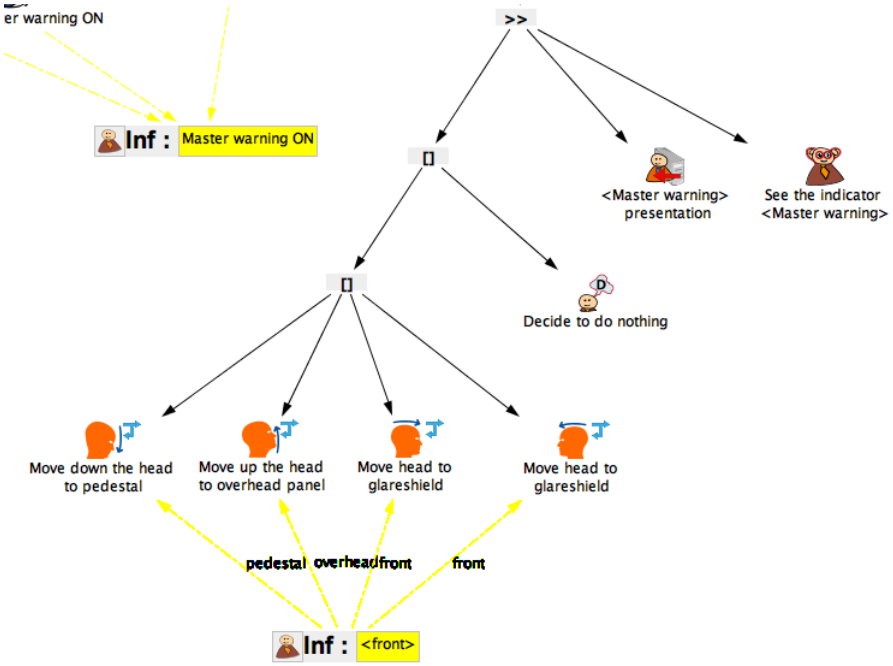


FIGURE 3.12 – Composant de la tâche du pilote qui voit un indicateur

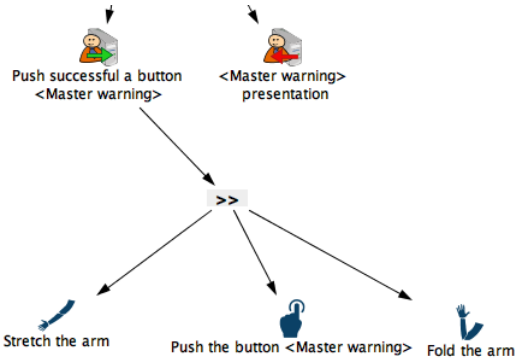


FIGURE 3.13 – Tâche d'appui sur le Master Warning

Ensuite, une fois le feu moteur détecté il est temps de le déclarer. Pour

cela, une tâche utilisateur abstraite a été définie. Elle est décomposée en une séquence de tâches (voir figure 3.14). Le pilote vérifie que le bouton "Eng" soit en état ON sur le panneau plafond. Cette dernière étant elle-même divisée en séquence suivantes : voir un indicateur avec comme paramètres le bouton "Eng1" et comme position le "panneau plafond". Ensuite le feu est déclaré et une séquence est effectuée : le pilote déclare le début de la procédure ECAM et annonce le feu moteur à l'ATC [2].

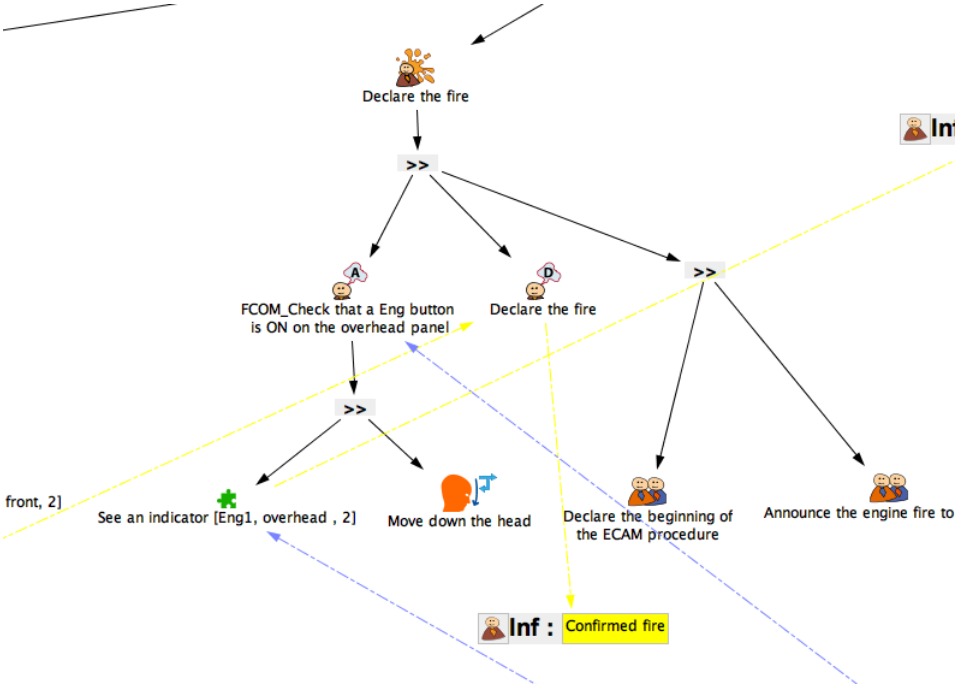


FIGURE 3.14 – Tâche abstraite de déclaration de feu moteur

Une fois le feu détecté, le pilote fixe l'alarme (voir figure 3.15). Cette tâche commence par l'extinction du moteur concerné par l'incendie. Ensuite, le système commence un compte à rebours de 10 secondes où le pilote peut choisir de compter en même temps ou de ne rien faire. Quand le compte à rebours est terminé et que le système affiche qu'il l'est, l'utilisateur le perçoit. À ce moment là, il prend la décision d'armer l'agent et il arme l'agent, cette tâche est abstraite.

2. Air Traffic Control, contrôle du trafic aérien, c'est un service de la circulation aérienne.

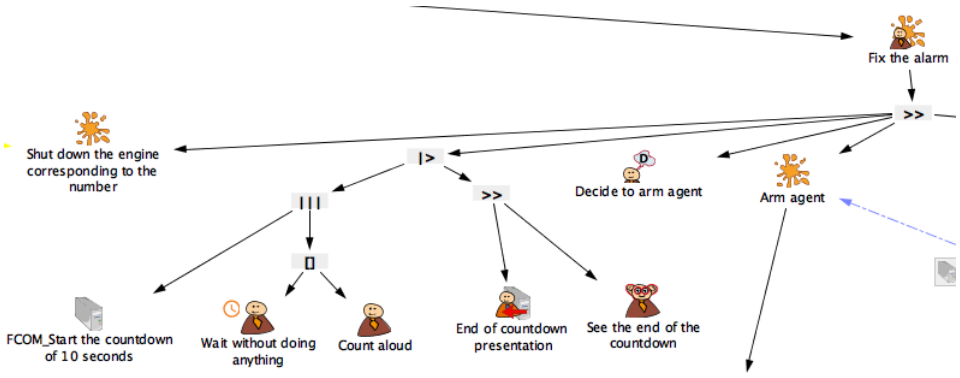


FIGURE 3.15 – Tâche abstraite de traitement de l'alarme

Comme on le voit en figure 3.16, la tâche abstraite "armer l'agent" est décomposable en une séquence constituée d'un composant et de tâches. Ce premier composant est "Voir un indicateur" que nous avons déjà rencontré précédemment mais prenant cette fois en paramètres "Eng1" et "Panneau plafond". Après cela, le pilote peut relever la protection de manière répétée jusqu'à ce qu'il arrive à un état de protection relevée et qu'il le perçoive.

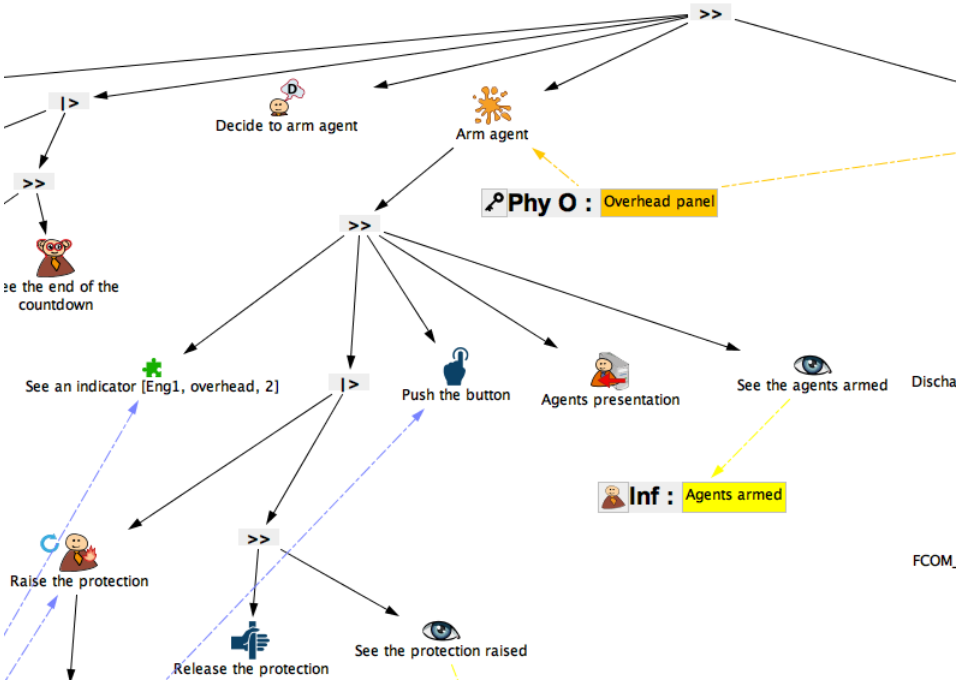


FIGURE 3.16 – Partie de tâches de traitement de l'alarme - Armer les agents

La figure 3.17 montre bien que relever la protection consiste en étendre le bras jusqu'au panneau plafond, prendre la protection et monter la protection. Une fois la protection relevée, le pilote appuie sur le bouton, les agents sont présentés ("agents presentation") par le système comme étant en état "SQUIB" (prêt à décharger) et perçus par le pilote.

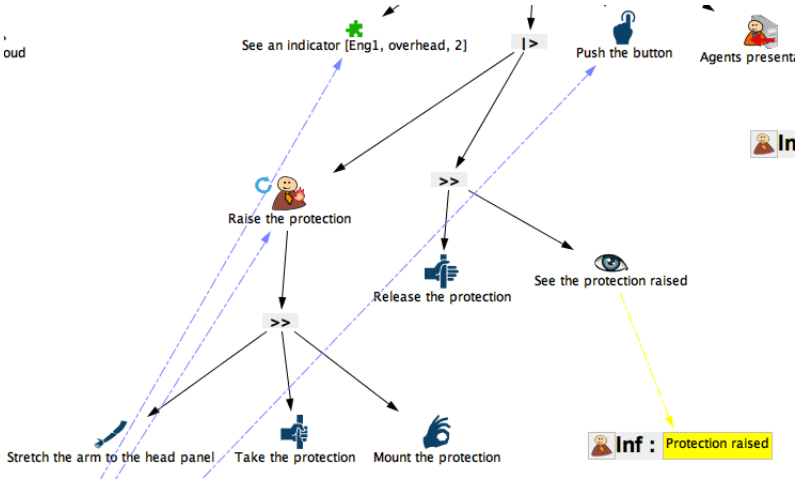


FIGURE 3.17 – Partie de tâches de armer les agents - enlever la protection

Vu que les agents sont prêts à être déchargés, le pilote peut passer à la tâche de décharger un agent qui est un composant dont on peut voir la décomposition en figure 3.18. La première tâche est un composant dont l'instanciation est effectuée pour le premier agent en figure 3.19 [3].

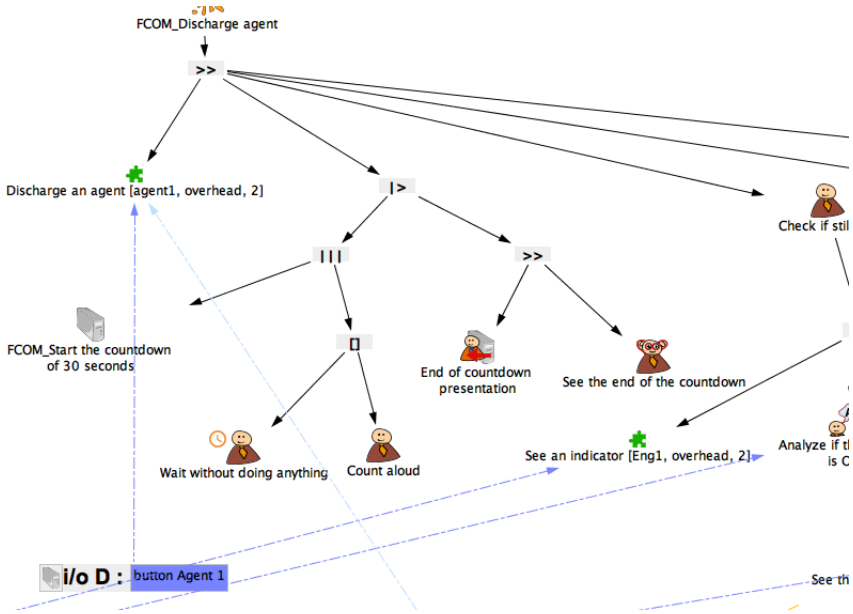


FIGURE 3.18 – Décomposition de la tâche abstraite de déchargement de l'agent ou des agents

Lorsque le déchargement de l'agent est terminé, le système fait un compte à rebours de 30 secondes où il peut compter en même temps ou ne rien faire

3. Lorsque le pilote déchargera le deuxième agent au besoin, l'instanciation sera en tout point identique, mis à part que le bouton ne sera plus agent1 mais bien agent2

jusqu'à ce que le système affiche la fin du compte à rebours et que le pilote le perçoive.

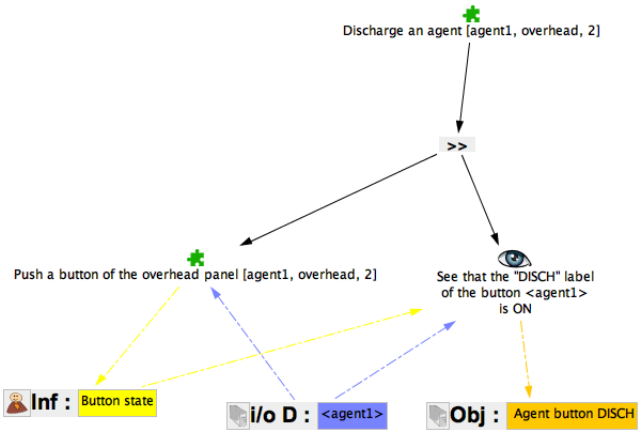


FIGURE 3.19 – Instance du composant décharger agent pour le bouton agent1

Une fois cela terminé, le pilote vérifie que le feu est bien éteint (figure 3.20). Pour cela, on définit une séquence de tâches qui constitue en un composant pour voir un indicateur (qui est le bouton du moteur qui était en feu EngX) ce qui lui fait relever la tête, une analyse afin de voir si ce bouton est toujours allumé, soit il l'est soit il ne l'est pas, le pilote redescend la tête du panneau plafond. Á partir de là, il prend la décision d'armer ou non le second agent. S'il doit le décharger, cela se passera de la même manière que pour le premier.

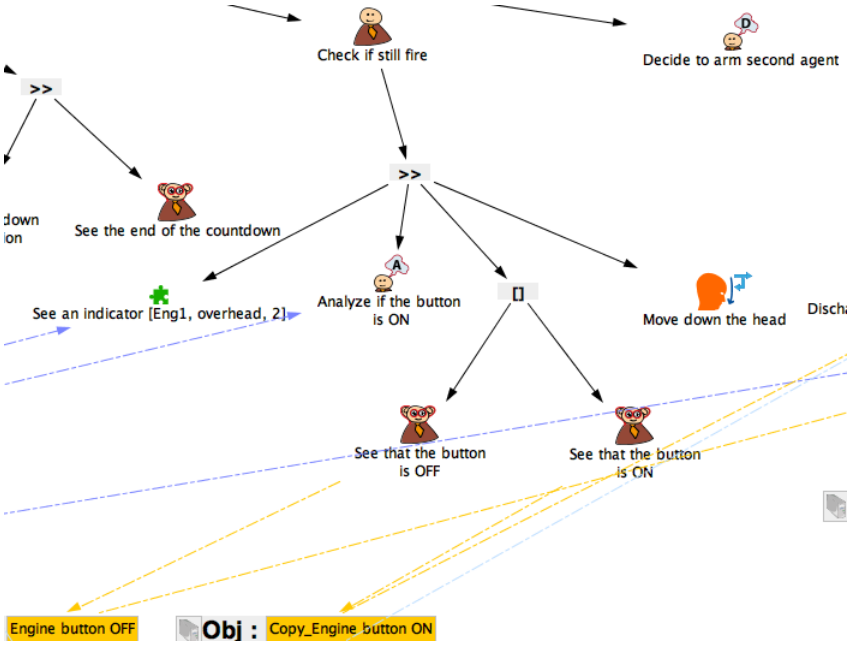


FIGURE 3.20 – Vérification de la présence du feu moteur

Une fois le feu moteur éteint, le pilote déclare la fin du feu moteur à son co-pilote, à l'ATC et revient à la tâche qu'il faisait avant l'incident.

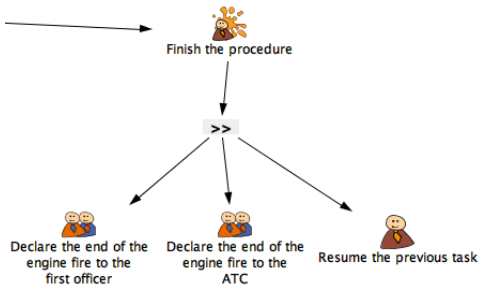


FIGURE 3.21 – Clôture de la procédure

1.3 GoPro

Afin de pouvoir établir une nouvelle interaction qui serait similaire, suivant nos définitions à la section 4, à l'interaction physique sans pour autant repartir de zéro, il a été décidé de partir d'une interaction bien connue et répandue qui est celle de déverrouillage de la GoPro. L'interaction de la GoPro se déroule comme suit :

- L'utilisateur effectue un "tap" sur l'écran,
- La fonctionnalité de déverrouillage de l'écran apparaît,
- L'utilisateur appuie sur le rond rouge,
- L'utilisateur maintient la pression sur le rond rouge et le glisse jusqu'à l'icone de "lock",
- Arrivé sur cette icone, l'utilisateur reste appuyé quelques millisecondes,
- L'écran se déverrouille.

FIGURE 3.22 – Fonction de déverrouillage de la GoPro Hero 4

À partir de cette interaction, nous allons dériver grâce à un ensemble d'étape vers un prototype réaliste de notre fonctionnalité.

1.4 Prototype

1.4.1 Étape 0 - Prototype avec la GoPro seule

Cette section a pour but d'illustrer l'interaction de la GoPro au travers de la réalisation d'un prototype.

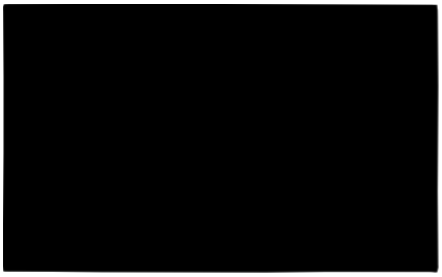


FIGURE 3.23 – Première étape de l'interaction avec la GoPro - Écran noir

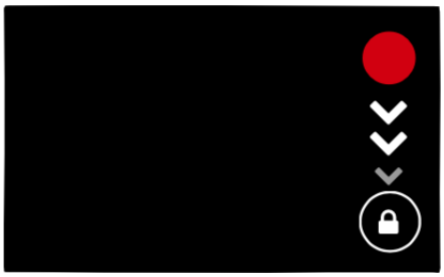


FIGURE 3.24 – Deuxième étape de l'interaction avec la GoPro - l'écran apparait avec le bouton rouge, le lock et les chevrons

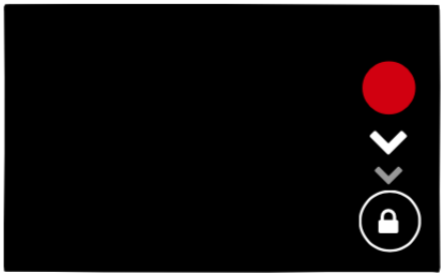


FIGURE 3.25 – Troisième étape de l'interaction avec la GoPro - l'utilisateur descend le bouton rouge et les chevrons disparaissent

FIGURE 3.26 – Quatrième étape de l'interaction avec la GoPro - l'utilisateur descend le bouton rouge et les chevrons disparaissent

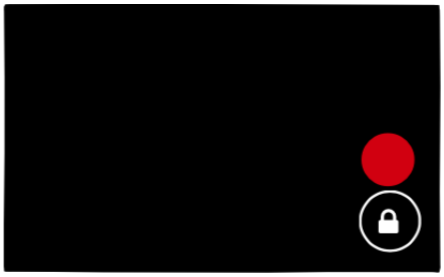


FIGURE 3.27 – Troisième étape de l'interaction avec la GoPro - l'utilisateur descend le bouton rouge et les chevrons ont disparus

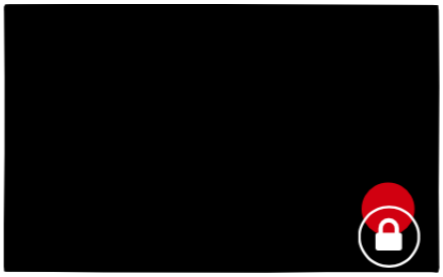


FIGURE 3.28 – Quatrième étape de l'interaction avec la GoPro - l'utilisateur descend le bouton rouge et le bouton passe sur l'arrière du lock

FIGURE 3.29 – Troisième étape de l'interaction avec la GoPro - l'utilisateur descend le bouton rouge et arrive derrière le lock

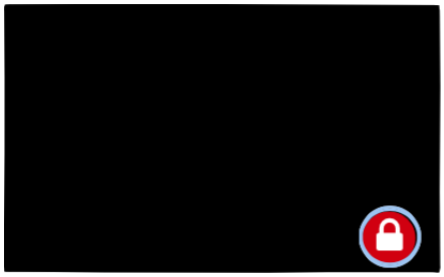


FIGURE 3.30 – Quatrième étape de l'interaction avec la GoPro - l'utilisateur reste appuyé sur le lock, la temporisation commence

FIGURE 3.31 – Troisième étape de l'interaction avec la GoPro - l'utilisateur reste appuyé sur le lock, la temporisation continue

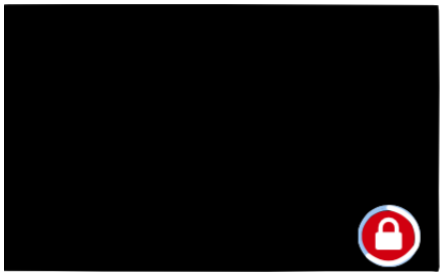


FIGURE 3.32 – Quatrième étape de l'interaction avec la GoPro - l'utilisateur reste appuyé sur le lock, la temporisation continue

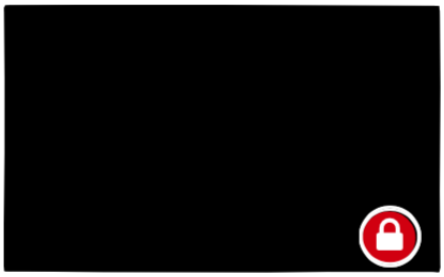


FIGURE 3.33 – Troisième étape de l'interaction avec la GoPro - l'utilisateur reste appuyé sur le lock, la temporisation est finie

FIGURE 3.34 – Quatrième étape de l'interaction avec la GoPro - l'écran est déverrouillé

1.4.2 Étape 7 - Prototype avec l'ajout des boutons agents

Pour arriver à ce prototype plusieurs étapes ont été nécessaires. Le processus pour passer d'une étape à une autre se base sur une question importante, est-ce qu'on est assez similaire par rapport à l'interface du panneau plafond physique existant ? Chaque étape consistait à se rapprocher au maximum de l'interface existante afin d'atteindre un niveau de similarité le plus haut possible tant au niveau interface qu'au niveau interaction. L'ensemble des prototypes relatif à ces étapes sont en annexe A. Les étapes sont :

- Adaptation de l'interaction à un écran plus grand (par rapport à celui de la GoPro),
- Adaptation du nombre de chevrons ainsi que de la taille,
- Changement du sens de l'interaction (on est passé de haut vers bas à bas vers haut) pour mieux correspondre à l'action physique qui est de remonter le clapet,
- Switch des icônes pour correspondre au point précédent,
- Ajout du texte avec une police "Airbus like" ainsi qu'une taille adaptée à celle qui se trouve dans le cockpit physique dans le bouton correspondant au "fire button",
- Changement de la couleur de fond pour imiter le fond physique,
- Changement de forme des boutons et de leur taille pour avoir un "Airbus like" et adaptation des chevrons,
- Changement de position des boutons de la droite vers la gauche toujours dans l'esprit d'imiter le panneau physique,
- Ajout des boutons de type agent.

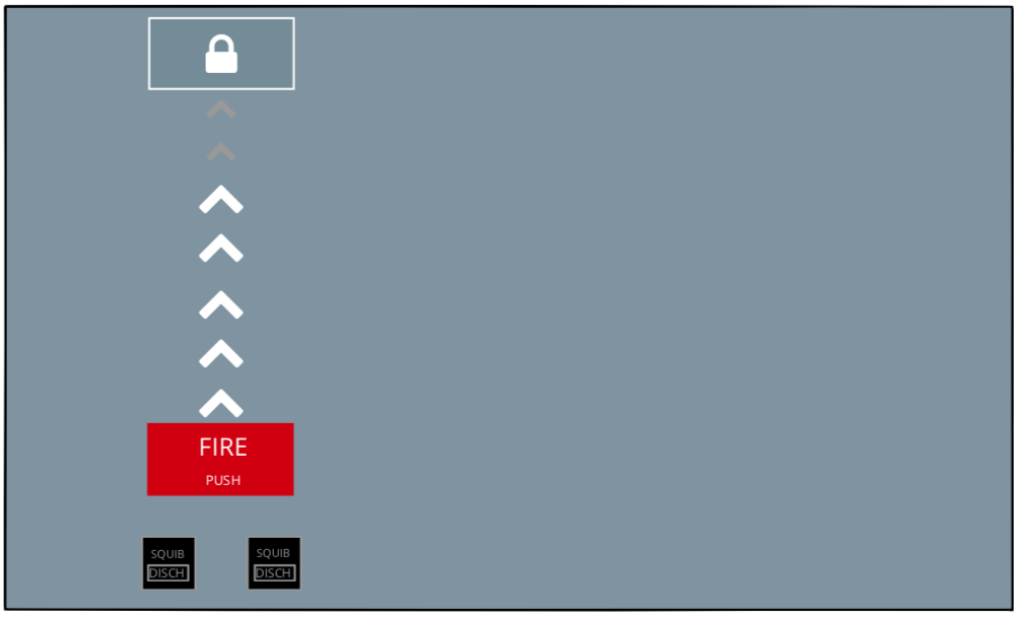


FIGURE 3.35 – Première étape de l'interaction avec le prototype où les agents ont été ajoutés

FIGURE 3.36 – Deuxième étape de l'interaction avec le prototype où les agents ont été ajoutés - L'utilisateur appuie sur le "fire button" et le glisse vers le haut, les chevrons disparaissent

FIGURE 3.37 – Troisième étape de l'interaction avec le prototype où les agents ont été ajoutés - L'utilisateur appuie sur le "fire button" et le glisse vers le haut, les chevrons disparaissent

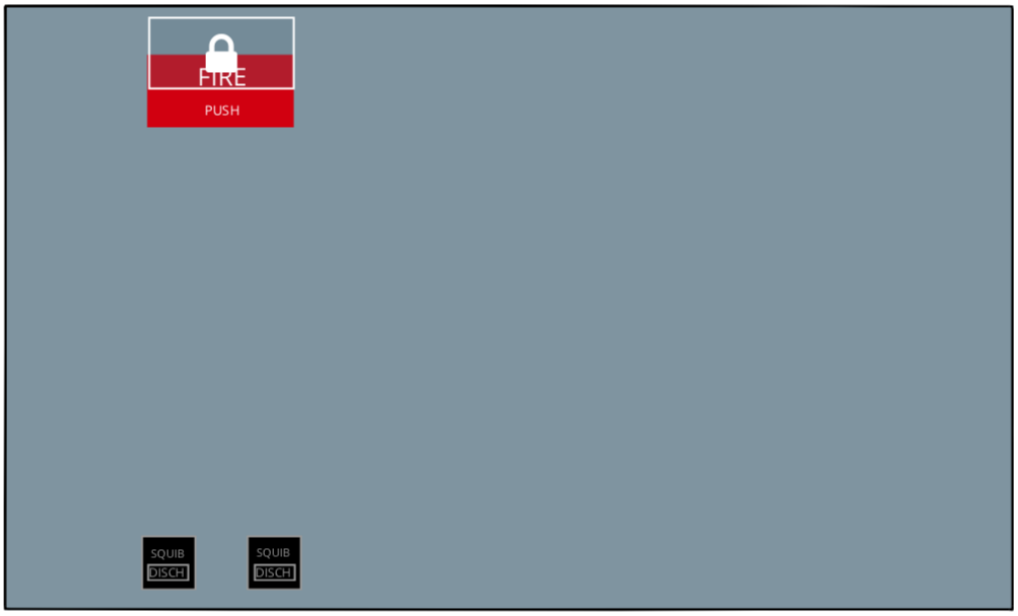


FIGURE 3.38 – Quatrième étape de l'interaction avec le prototype où les agents ont été ajoutés - L'utilisateur glisse le "fire button" sur l'icone de "lock" ce qui le fait passer derrière

FIGURE 3.39 – Cinquième étape de l'interaction avec le prototype où les agents ont été ajoutés - Le "fire button" est complètement derrière l'icone de lock, la temporisation commence

FIGURE 3.40 – Sixième étape de l'interaction avec le prototype où les agents ont été ajoutés - Le "fire button" est complètement derrière l'icone de lock, la temporisation continue

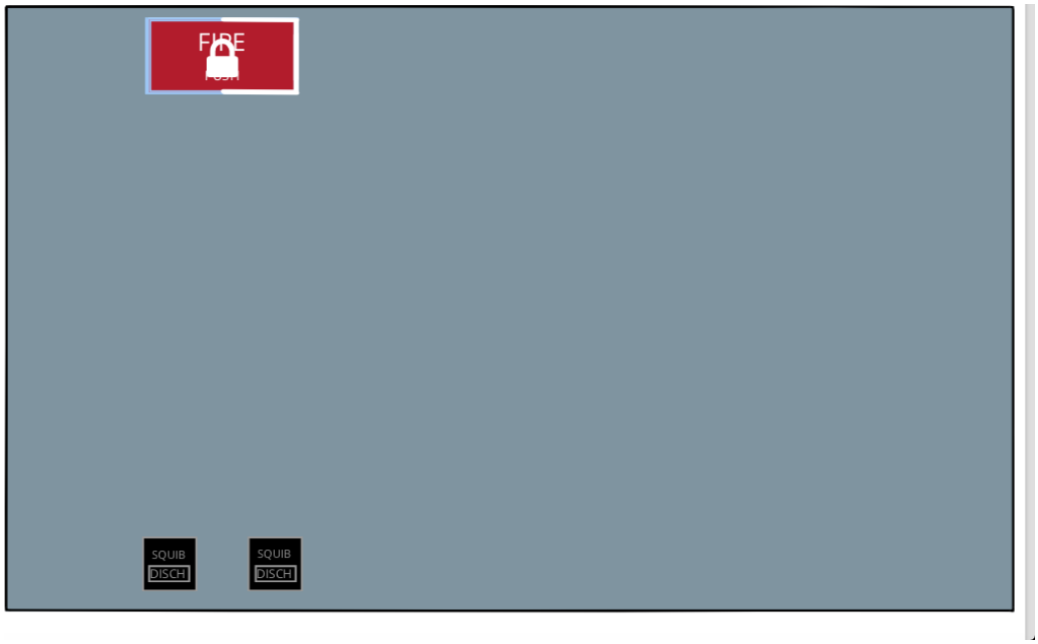


FIGURE 3.41 – Septième étape de l'interaction avec le prototype où les agents ont été ajoutés - Le "fire button" est complètement derrière l'icone de lock, la temporisation continue

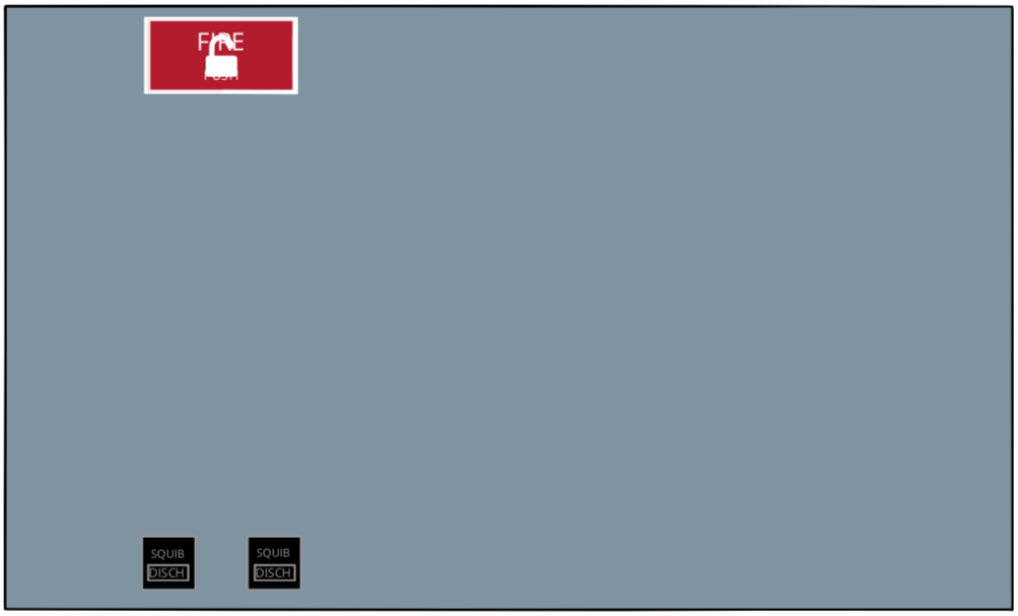


FIGURE 3.42 – Huitième étape de l'interaction avec le prototype où les agents ont été ajoutés - Le "fire button" est complètement derrière l'icone de lock, la temporisation est finie et le verrou est déverrouillé

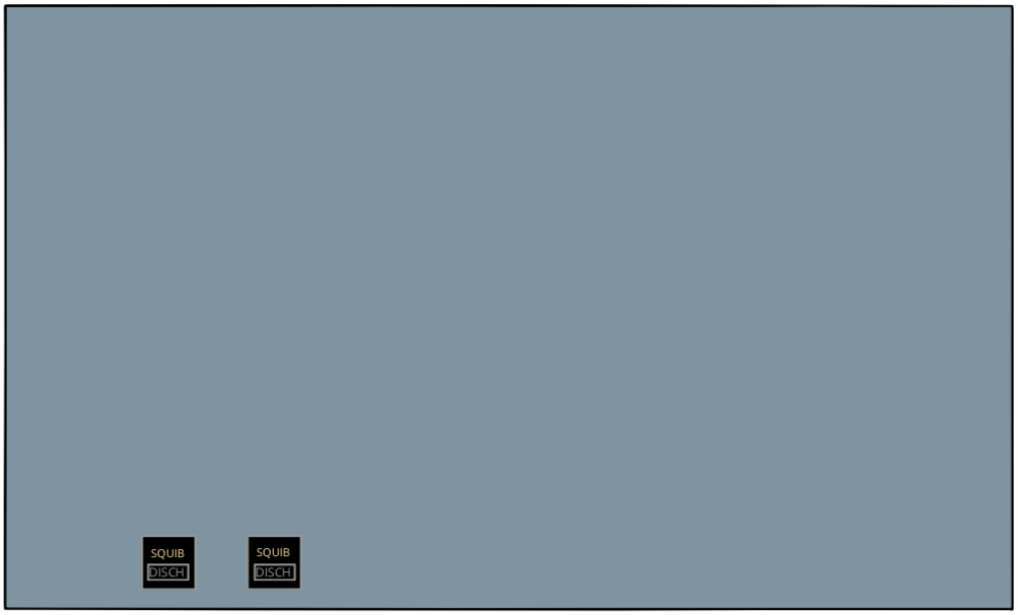


FIGURE 3.43 – Neuvième étape de l'interaction avec le prototype où les agents ont été ajoutés - Le "fire button" disparaît et les agents sont passés à l'état "SQUIB" (prêt à décharger)

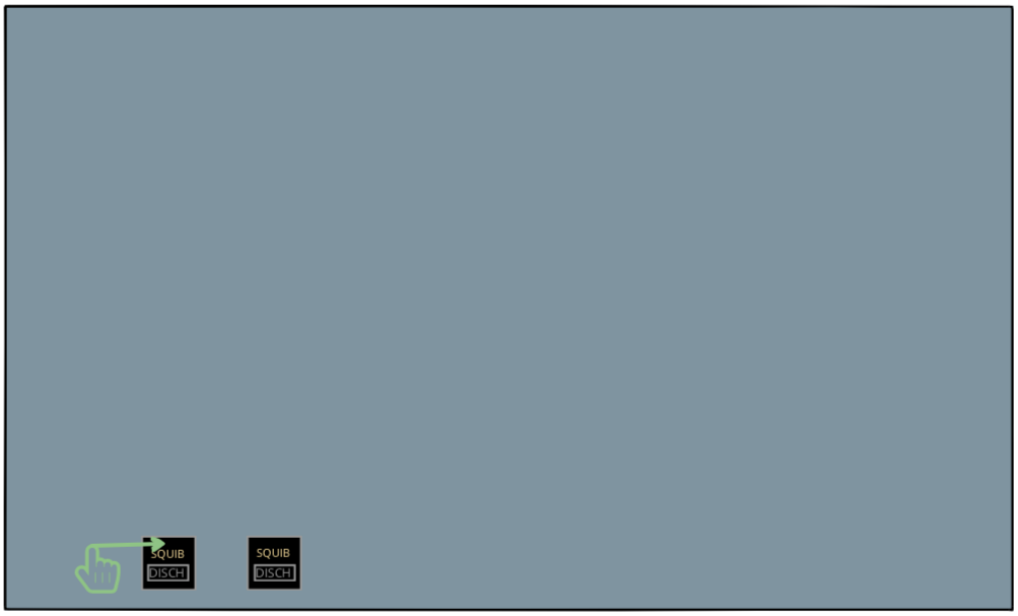


FIGURE 3.44 – Dixième étape de l'interaction avec le prototype où les agents ont été ajoutés - L'utilisateur appuie sur le premier agent

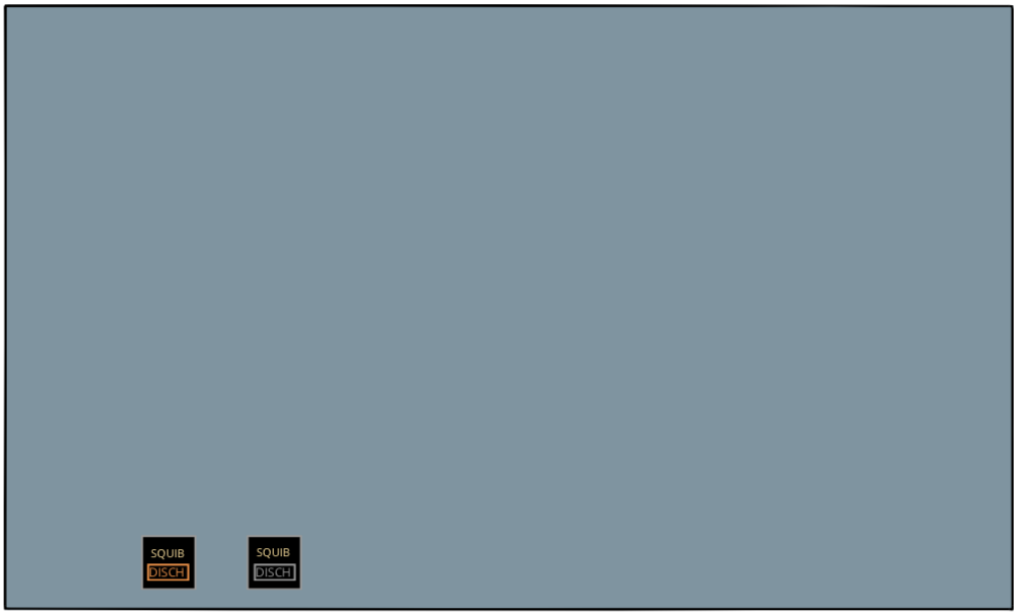


FIGURE 3.45 – Onzième étape de l'interaction avec le prototype où les agents ont été ajoutés - L'agent 1 est dans l'état "déchargé"

1.4.3 Étape 11 - Prototype final

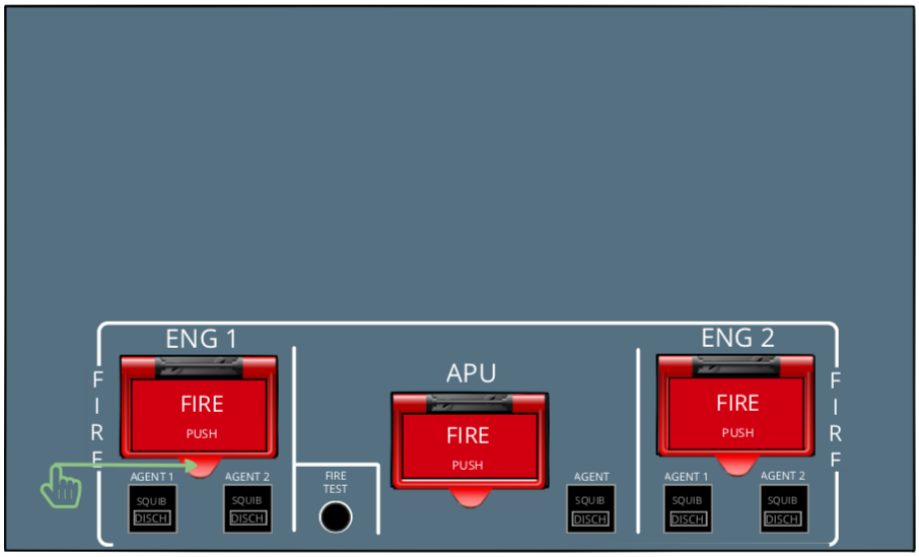


FIGURE 3.46 – Première étape de l'interaction avec le prototype final - L'utilisateur appuie sur le clapet à l'endroit prévu à cet effet

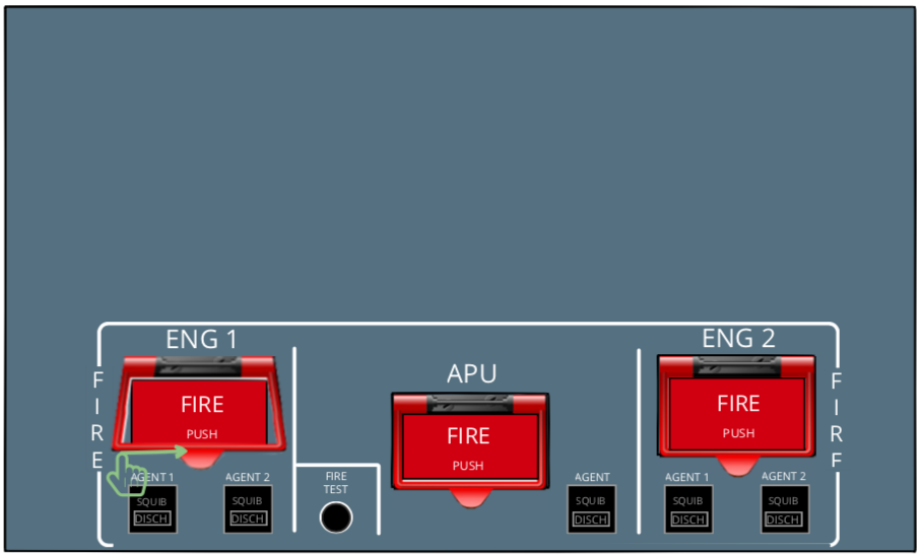


FIGURE 3.47 – Deuxième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

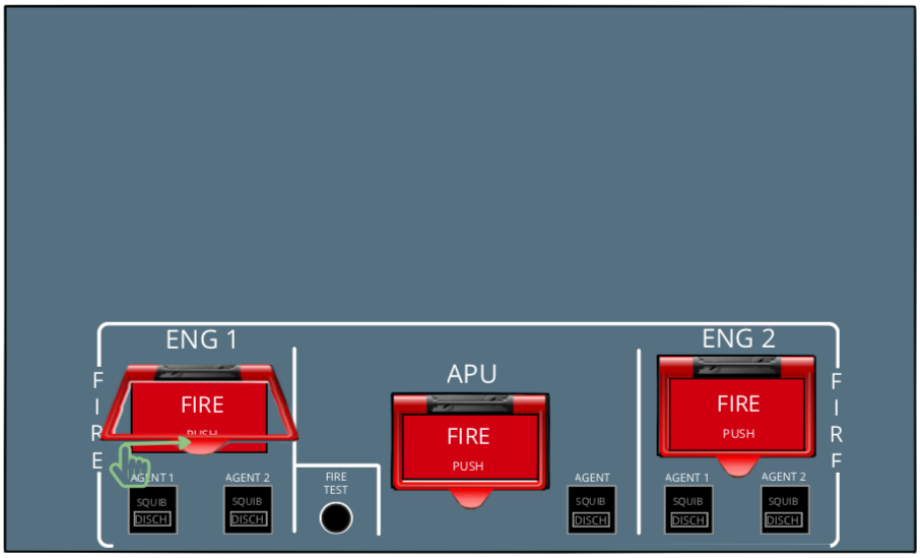


FIGURE 3.48 – Troisième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

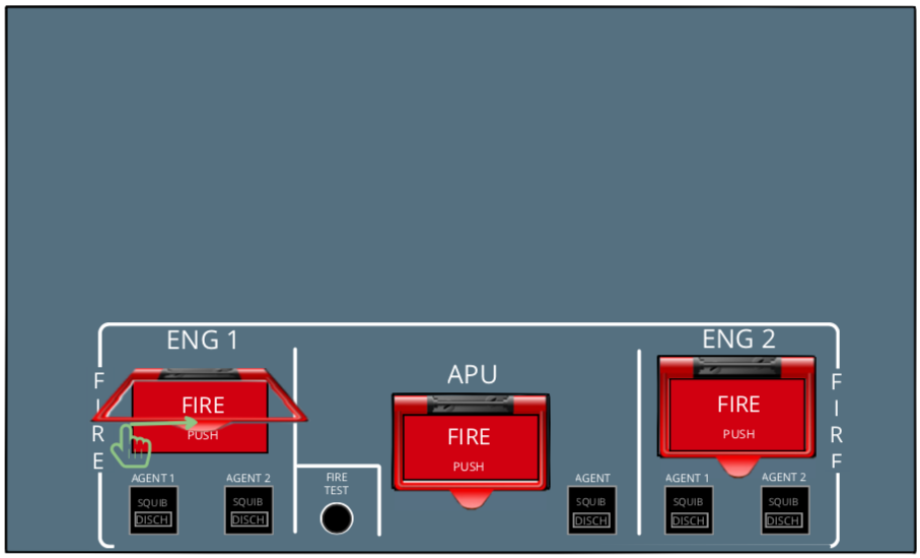


FIGURE 3.49 – Troisième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

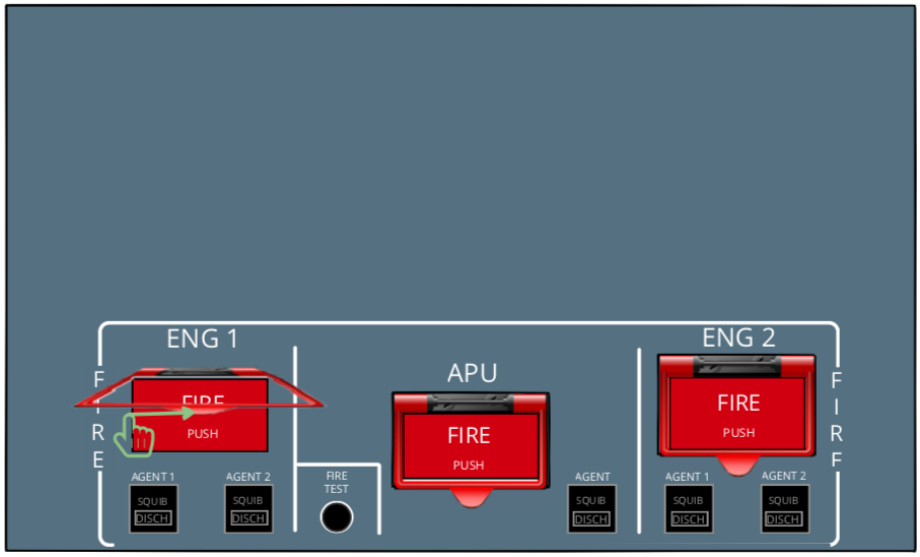


FIGURE 3.50 – Quatrième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

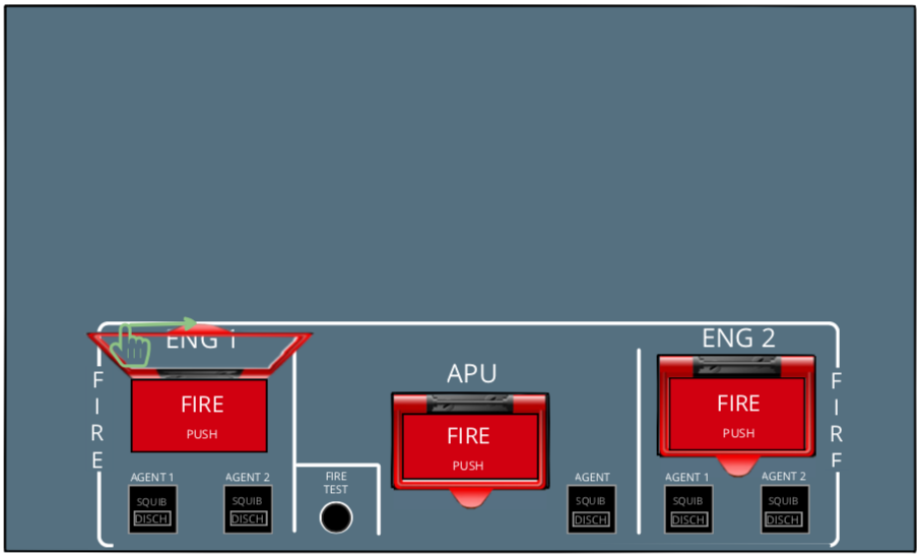


FIGURE 3.51 – Cinquième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

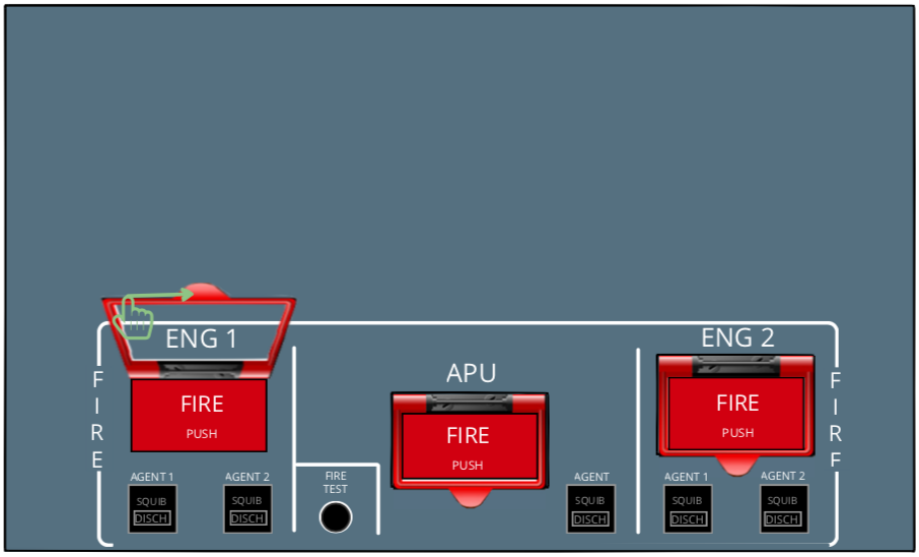


FIGURE 3.52 – Sixième étape de l'interaction avec le prototype final - L'utilisateur glisse le clapet vers le haut

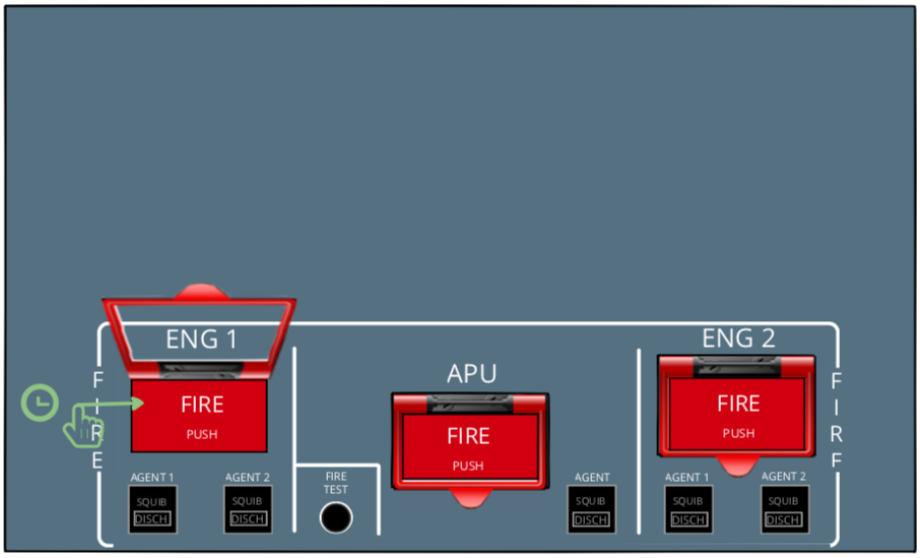


FIGURE 3.53 – Septième étape de l'interaction avec le prototype final - L'utilisateur appuie sur le "fire button", la temporisation commence sur ce même bouton

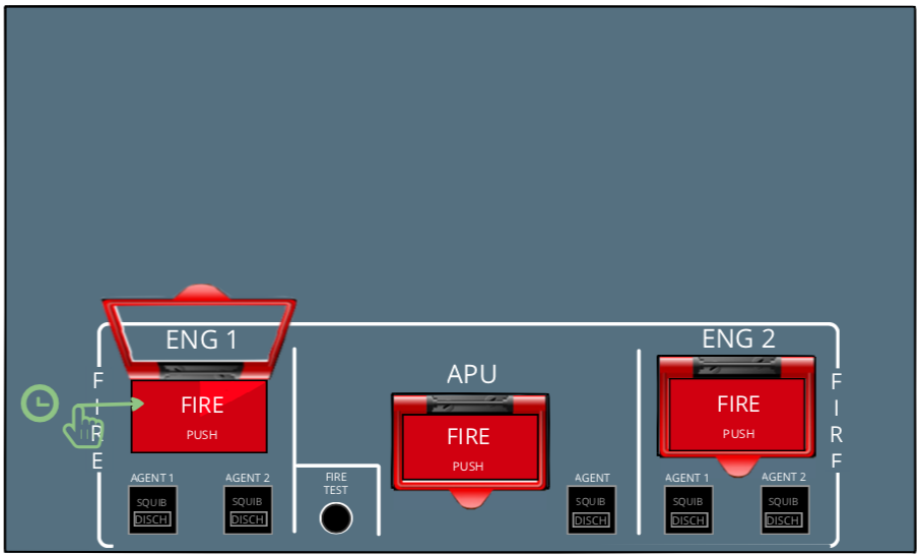


FIGURE 3.54 – Huitième étape de l'interaction avec le prototype final - L'utilisateur appuie sur le "fire button", la temporisation continue sur ce même bouton

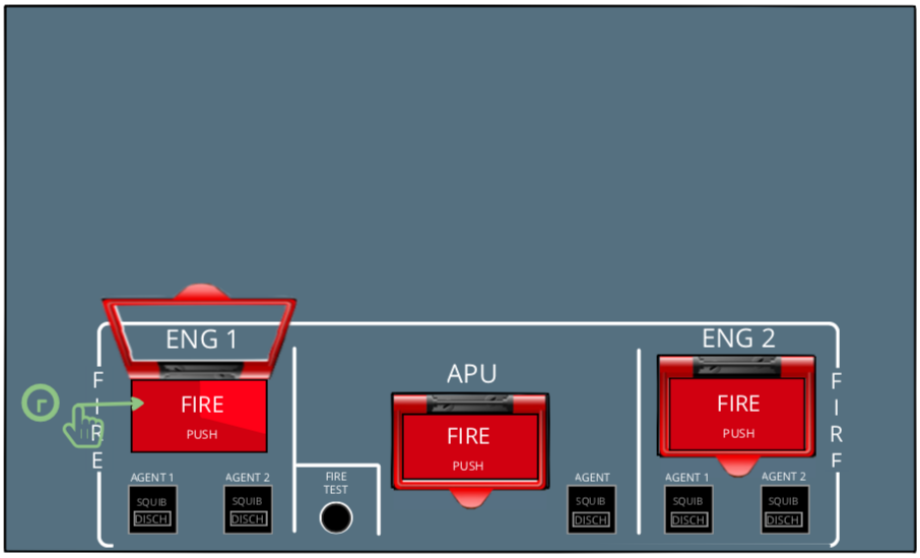


FIGURE 3.55 – Neuvième étape de l'interaction avec le prototype final - L'utilisateur appuie sur le "fire button", la temporisation continue sur ce même bouton

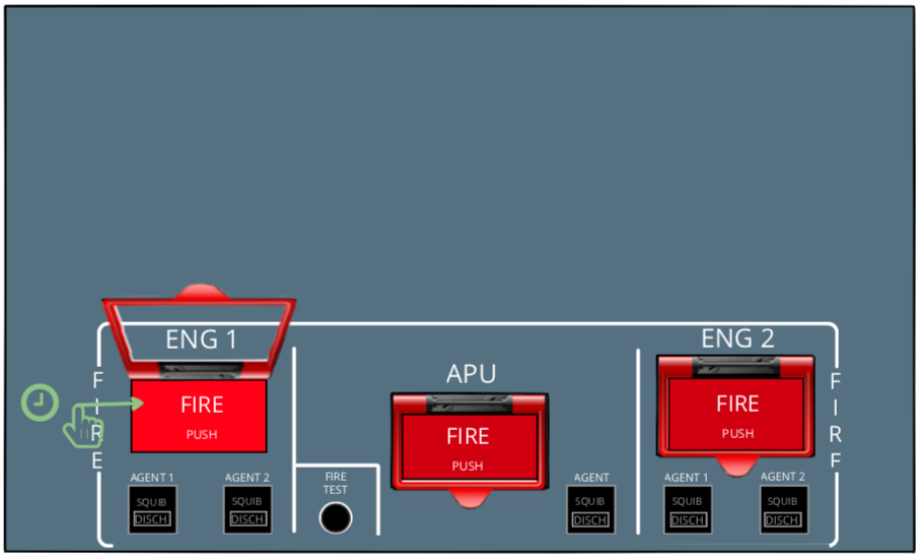


FIGURE 3.56 – Dixième étape de l'interaction avec le prototype final - L'utilisateur appuie sur le "fire button", la temporisation continue sur ce même bouton

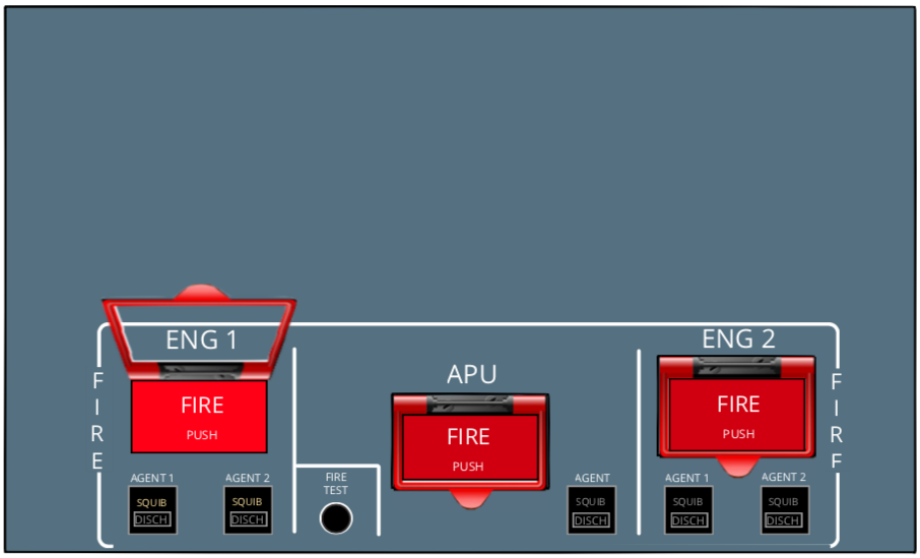


FIGURE 3.57 – Onzième étape de l'interaction avec le prototype final - L'utilisateur relâche sur le "fire button", la temporisation se termine sur ce même bouton, les agents sont passés à l'état "SQUIB", prêt à décharger

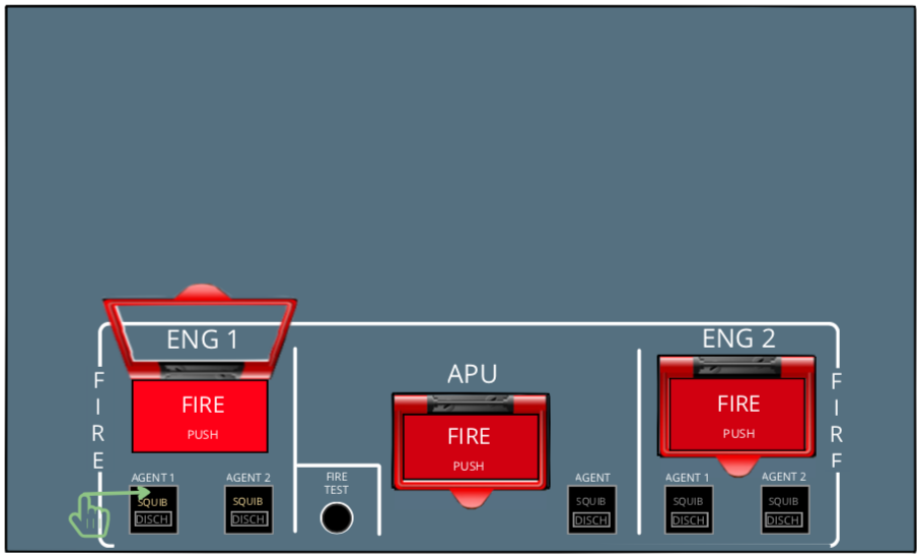


FIGURE 3.58 – Douzième étape de l'interaction avec le prototype final - L'utilisateur appuie sur le bouton agent1

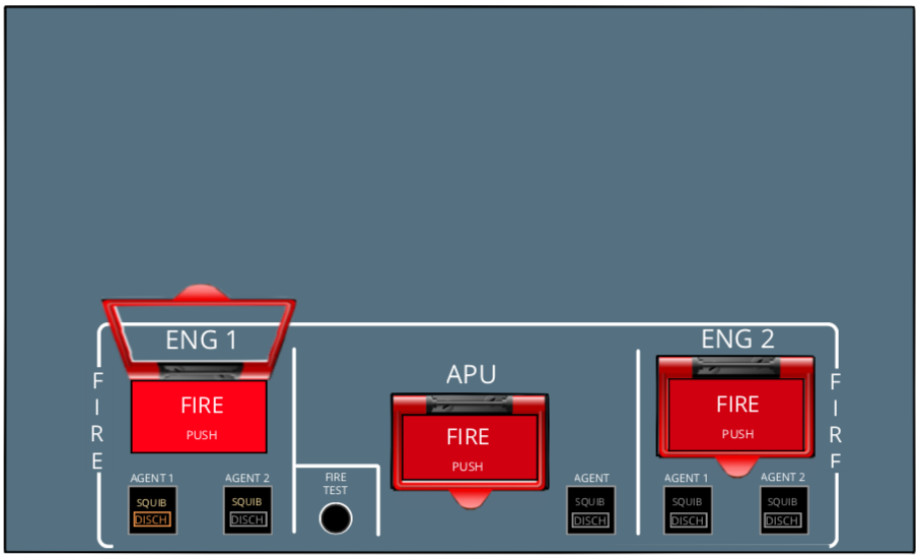


FIGURE 3.59 – Treizième étape de l'interaction avec le prototype final - Le bouton agent1 est passé à l'état déchargé

1.5 Analyse des modèles de tâches des prototypes

Pour tous les modèles, on considère que "Détecter l'alarme", "Confirmer l'alarme" et "Terminer l'alarme" restera tel quel. Ceci car nous n'intervenons pas sur le même device. En effet, vu que le premier prototype se passe sur la GoPro, nous n'aurons, par exemple, pas de "Terminer l'alarme" puisque, nous le verrons plus loin, le prototype ne correspond pas. De ce fait, pour chaque

prototype, nous garderons uniquement les morceaux de modèle qui change. Les modèles de tâches complet sont disponibles en annexe B.

1.5.1 Étape 0 - Prototype avec la GoPro seule

Pour ce modèle, nous avons pris le prototype (0) (section 1.4.1) qui est celui de la GoPro pur. L'interaction de ce prototype se déroule comme suit :

— L'utilisateur effectue un "tap" sur l'écran,
— La fonctionnalité de déverrouillage de l'écran apparaît,
— L'utilisateur appuie sur le rond rouge,
— L'utilisateur maintient la pression sur le rond rouge et le glisse jusqu'à l'icone de "lock",
— Arrivé sur cette icone, l'utilisateur reste appuyé quelques millisecondes,
— L'écran se déverrouille.

On peut le constater, il n'y a aucune correspondance avec le modèle de tâches de l'interaction physique avec le panneau plafond.

Dans le prototype GoPro (0), nous n'avons pas de bouton de type "FIRE" et donc aucune protection à relever. Ce prototype ne dispose pas non plus de boutons de type "AGENT".

Il contient uniquement un bouton rouge et une icône de lock dans un cercle. Par rapport à notre modèle de tâches de base, on pourrait garder les deux premières actions :

— "Eteindre le moteur qui correspond au numéro du bouton",
— "FCOM_commencer un décompte de 10 secondes".

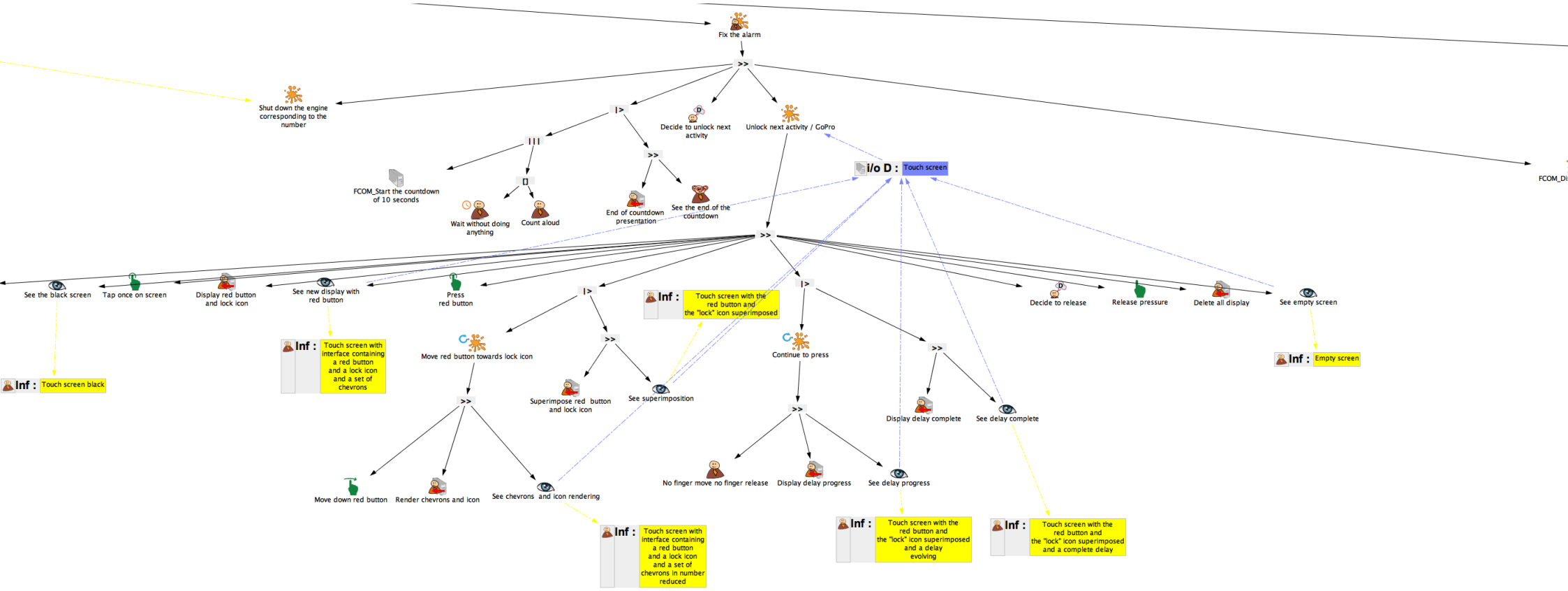


FIGURE 3.60 – Modèle de tâches correspondant à la GoPro

Mais, à partir de là la tâche "Armer agent" va être modifiée. Nous allons la modifier en fonction des étapes précédemment énoncées.

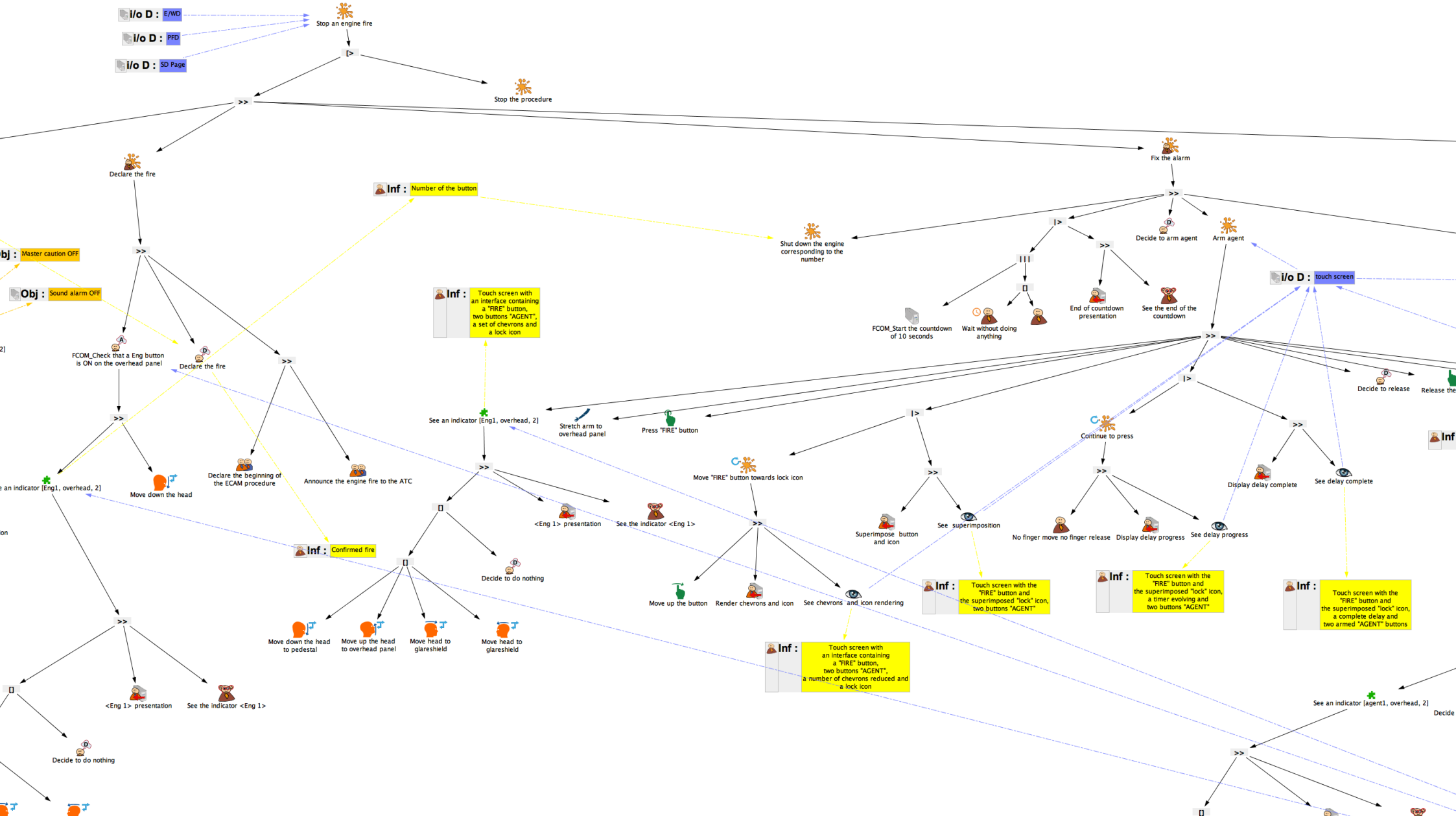
Pour la tâche "Décharger agent" dans le modèle de tâches, on peut dire que le prototype n'est pas compatible puisque, comme énoncé précédemment, celui-ci ne dispose pas de bouton de type "AGENT" et nous n'avons, par conséquent, aucun moyen de réaliser la tâche "Décharger agent".

Une alternative à ça serait de migrer de ce prototype vers un prototype plus "évolué" et plus "similaire" avec un bouton de type "FIRE" pour pouvoir armer l'agent et deux boutons de type "AGENT".

1.5.2 Étape 7 - Prototype avec l'ajout des boutons agents

Pour ce modèle, nous avons pris le prototype (7) (section 1.4.2) qui ressemble au véritable panneau puisqu'on a ajouté le bouton de type "FIRE" et les boutons de type "AGENT" tout en conservant un côté GoPro de par l'interaction. L'interaction pour armer les agents se déroule comme suit :

- Le système affiche directement l'interface,
- L'utilisateur appuie sur le bouton "FIRE" rouge,
- L'utilisateur maintient la pression sur le bouton et le glisse jusqu'à l'icone de "lock",
- Une fois qu'il est sur l'icone, l'utilisateur reste appuyé quelques millisecondes,
- Le bouton "FIRE" disparait et les agents sont armés.


i/o D : E/WD
i/o D : PFD
i/o D : SD Page
Stop an engine fire
[>
Stop the procedure
>>
Declare the fire
Fix the alarm
Inf : Number of the button
Shut down the engine corresponding to the number
Obj : Master caution OFF
Obj : Sound alarm OFF
FCOM_Check that a Eng button is ON on the overhead panel
Declare the fire
Inf : Touch screen with an interface containing a "FIRE" button, two buttons "AGENT", a set of chevrons and a lock icon
|>
Decide to arm agent
Arm agent
|||
[]
FCOM_Start the countdown of 10 seconds
Wait without doing anything
End of countdown presentation
See the end of the countdown
i/o D : touch screen
Decide to release
See an indicator [Eng1, overhead, 2]
Stretch arm to overhead panel
Press "FIRE" button
Move "FIRE" button towards lock icon
Continue to press
Move down the head
Declare the beginning of the ECAM procedure
Announce the engine fire to the ATC
<Eng 1> presentation
See the indicator <Eng 1>
Superimpose button and icon
See superimposition
Display delay complete
See delay complete
No finger move no finger release
Display delay progress
See delay progress
Inf : Confirmed fire
Decide to do nothing
Move up the button
Render chevrons and icon
See chevrons and icon rendering
Inf : Touch screen with the "FIRE" button and the superimposed "lock" icon, two buttons "AGENT"
Inf : Touch screen with the "FIRE" button and the superimposed "lock" icon, a timer evolving and two buttons "AGENT"
Inf : Touch screen with the "FIRE" button and the superimposed "lock" icon, a complete delay and two armed "AGENT" buttons
Move down the head to pedestal
Move up the head to overhead panel
Move head to glareshield
Move head to glareshield
Inf : Touch screen with an interface containing a "FIRE" button, two buttons "AGENT", a number of chevrons reduced and a lock icon
<Eng 1> presentation
See the indicator <Eng 1>
Decide to do nothing
See an indicator [agent1, overhead, 2]
Decide

L'interaction pour décharger les agents se passe comme suit :
- L'utilisateur appuie sur le premier bouton agent (ce prototype se lisant de gauche à droite),
- Le bouton passe en "déchargé" ("DISCH"),
- Si le feu moteur persiste, l'utilisateur répète les deux actions précédentes sur le deuxième bouton.

La différence majeur avec le prototype 0 est que ce prototype-ci (7) est compatible avec le modèle de tâches, cela se voit de suite : on peut armer les agents et les décharger.

La tâche "Armer agent" va reprendre plusieurs actions du prototype(0) comme le "maintenir le bouton rouge", "déplacer le bouton sur l'icône lock", etc.
Sauf qu'ici nous pouvons garder le "Regarder un indicateur" du modèle de tâches de l'interaction physique puisque nous avons bien un bouton "FIRE" comme dans le panneau physique.
Nous avons également changé le sens de déplacement du bouton rouge pour être plus "similaire" au panneau physique.

Le composant "Décharger un agent" restera identique à celui du panneau physique puisque nous regardons bien l'indicateur, ensuite nous appuyons sur le bouton "AGENT", revenons à notre position de départ et les boutons ont la même réaction.

En revanche, par rapport à l'interface physique nous sommes encore assez éloigné. En effet, nous n'avons pas deux actions bien distinctes mais bien une seule "longue" action : glisser et maintenir. Alors que l'interface physique demande une action pour relever le capot et une action pour appuyer sur le bouton "FIRE".

Une solution est d'ajouter un capot à notre bouton "FIRE" et de modifier l'interaction de style GoPro pour passer à une interaction plus "réaliste" et "similaire".

1.5.3 Étape 11 - Prototype final

Pour ce modèle, nous avons utilisé le prototype (11) (section 1.4.3) qui est le plus proche du panneau physique. Tous les composants présents dans le panneau physique (pour l'arrêt d'un feu moteur) sont présents dans ce prototype.

Nous avons un ajout majeur par rapport au prototype précédent (7) (section 1.4.2) qui aura un impact sur l'interaction, celui-ci est l'ajout du capot sur le bouton "FIRE". Nous avons également des ajouts au niveau de l'interface uniquement comme les différents labels, etc.

L'interaction pour armer les agents se passe donc comme suit :
- Le système affiche directement l'interface,
- L'utilisateur appuie sur le bord du capot (prévu à cet effet),
- L'utilisateur glisse son doigt vers le haut pour ouvrir le capot,
- L'utilisateur effectue cette action jusqu'à sentir une vibration lui indiquant que le capot est relevé à son maximum,
- L'utilisateur relâche sa pression,
- L'utilisateur appuie sur le bouton "FIRE" et maintient la pression,

— Le système effectue une temporisation directement sur le bouton [4],
— Une fois cette temporisation complétée, l'utilisateur retire son doigt après avoir eu une vibration signalant la fin de la temporisation ;

L'interaction pour déclencher l'agent reste semblable au prototype précédent (7) :

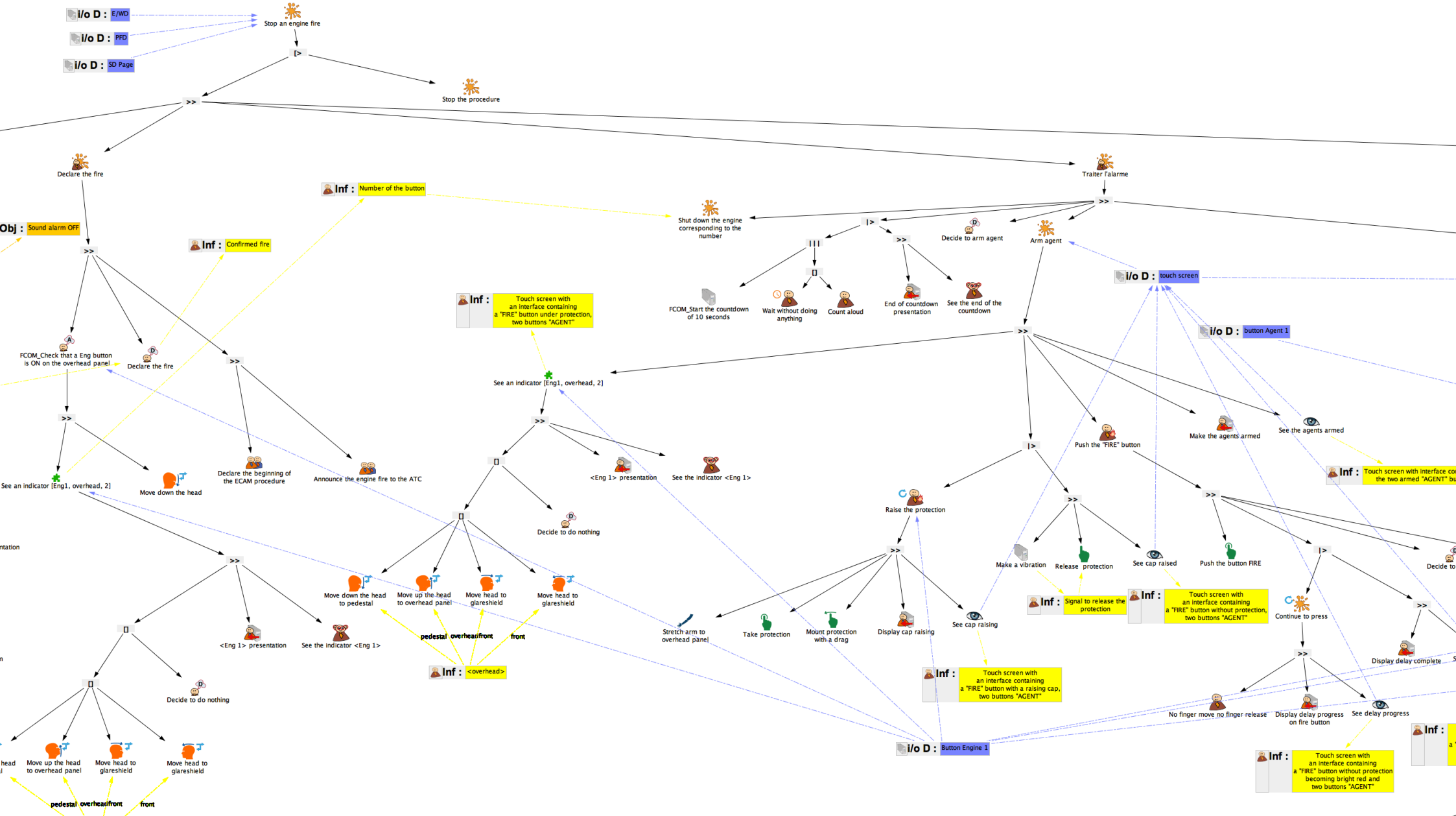
— L'utilisateur appuie sur le premier bouton agent (ce prototype se lisant de gauche à droite),
— Le bouton passe en "déchargé" ("DISCH"),
— Si le feu moteur persiste, l'utilisateur répète les deux actions précédentes sur le deuxième bouton.

La tâche "Armer agent" va reprendre plusieurs actions du modèle de tâches de l'interaction physique comme la tâche abstraite "Relever la protection" qui sera légèrement modifiée pour correspondre au tactile, la tâche "Appuyer sur le bouton" restera sous la forme tactile également (appui long). Nous avons donc bien deux actions distinctes pour réaliser ces actions. Contrairement au prototype précédent (7) où nous faisons ces actions en une seule action suivie.

Le composant "Décharger un agent" restera identique à celui du modèle de tâches pour le panneau physique.

Nous avons ici une interface qui se compose comme le panneau physique, des interactions qui utilisent la même "séquence" d'actions : on relève la protection et on appuie sur le bouton.

4. Le bouton a ses quartiers qui prennent une couleur rouge vif et ce jusqu'à ce que tous les quartiers du bouton soient rouge vif


i/o D : E/WD
i/o D : PFD
i/o D : SD Page
Stop an engine fire
[>
Stop the procedure
>>
Declare the fire
Obj : Sound alarm OFF
Inf : Number of the button
Inf : Confirmed fire
FCOM_Check that a Eng button is ON on the overhead panel
Declare the fire
See an indicator [Eng1, overhead, 2]
Move down the head
Declare the beginning of the ECAM procedure
Announce the engine fire to the ATC
Inf : Touch screen with an interface containing a "FIRE" button under protection, two buttons "AGENT"
See an indicator [Eng1, overhead, 2]
[]
<Eng 1> presentation
See the indicator <Eng 1>
Decide to do nothing
Move down the head to pedestal
Move up the head to overhead panel
Move head to glareshield
Move head to glareshield
pedestal
overhead
front
Inf : <overhead>
<Eng 1> presentation
See the indicator <Eng 1>
Decide to do nothing
Move up the head to overhead panel
Move head to glareshield
Move head to glareshield
pedestal
overhead
front
Traiter l'alarme
Shut down the engine corresponding to the number
|||
FCOM_Start the countdown of 10 seconds
Wait without doing anything
Count aloud
|>
Decide to arm agent
Arm agent
End of countdown presentation
See the end of the countdown
i/o D : touch screen
i/o D : button Agent 1
Push the "FIRE" button
Make the agents armed
See the agents armed
Inf : Touch screen with interface co
the two armed "AGENT" bu
Raise the protection
Make a vibration
Release protection
See cap raised
Push the button FIRE
Inf : Signal to release the protection
Inf : Touch screen with an interface containing a "FIRE" button without protection, two buttons "AGENT"
Stretch arm to overhead panel
Take protection
Mount protection with a drag
Display cap raising
See cap raising
Inf : Touch screen with an interface containing a "FIRE" button with a raising cap, two buttons "AGENT"
Continue to press
Display delay complete
No finger move no finger release
Display delay progress on fire button
See delay progress
Inf : Touch screen with an interface containing a "FIRE" button without protection becoming bright red and two buttons "AGENT"
Inf :
i/o D : Button Engine 1

1.6 Modèles systèmes et modèles de tâches

Dans cette section, le but n'est pas d'entrer dans les détails au niveau des modèles systèmes. En effet, le but ici est de montrer qu'il est possible de faire une correspondance entre les modèles PetShop et les modèles de tâches.

En utilisant l'outil de correspondance qui met en relation les modèles Hamsters et Petshop, il est possible de mettre en correspondance une tâche interactive d'entrée du côté du modèle de tâches et un "eventHandler" [5]du côté de Petshop ainsi qu'une tâche d'output avec un "stateholder" [6]

Ainsi, nous nous assurons que nous avons couvert tous les cas dans notre modèle de tâches ainsi que dans le système même comme on peut le voir en figure 3.63.

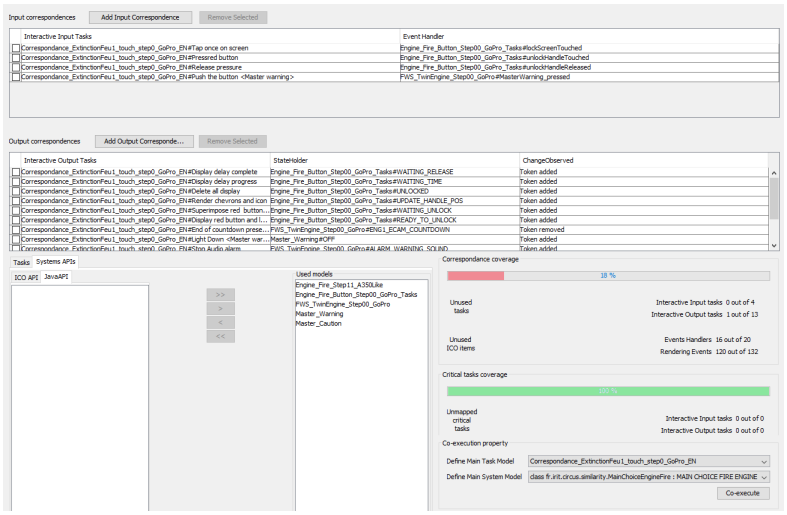


FIGURE 3.63 – Correspondance entre les modèles Petshop et les modèles Hamsters

5. un eventHandler étant un évènement qui permet de passer d'un stateholder à un autre
6. un stateholder étant un état dans lequel se trouve le système.

Chapitre 4

Génération d'une formation

L'idée de ce chapitre est de montrer un outil, une application qui permettrait à un utilisateur de générer une formation automatiquement à partir d'un modèle Hamsters ainsi que le besoin que cette application comble.

1 Besoin

Après avoir étudié l'interface physique et défini une nouvelle interaction avec notre interface tactile, une question est survenue. Comment former le personnel qui est habitué à son interface actuelle à chaque nouvelle interface en un temps minime ?

2 Processus

Après avoir émis ce besoin et après réflexion, l'idée de départ est de prendre le fichier XML que fournit l'application Hamsters, qui représente la modélisation de tâches sous forme d'arbre XML, et d'en faire une application.
À partir de là, une analyse du fichier XML a du être faite. Pour commencer, un simple parcours du fichier XML a été programmé. De là, nous avons pu nous rendre compte de deux choses : le fichier contient beaucoup de données parasites, le fichier dispose bien des tâches utilisateurs mais dans un ordre qui est incorrect. Deux solutions sont apportées :

1. Passage sur l'entièreté du fichier et nous gardons uniquement ce dont nous avons besoin
2. Remise dans l'ordre des tâches via les coordonnées X,Y qui représente l'ordre dans la visualisation du modèle de tâches. [1]

De ces solutions a émergé une question : fait-on tout cela dans le fichier déjà existant ou est-ce qu'on en crée un nouveau ? L'option qui a été choisie est la seconde. Nous avons créé un nouveau fichier afin de ne garder que ce que nous souhaitions et de garder la base intacte. En effet, nous verrons plus tard des améliorations possibles de cette application qui pourraient impliquer que nous devions garder le fichier de base pour un traitement ou l'autre (section 1 du chapitre 5).

Pour la première solution citée plus haut, l'application

— passe sur chaque ligne du fichier XML,

1. Un modèle de tâche se lit du haut vers le bas et de la gauche vers la droite.

— enlève les données inutiles pour nous et garde les utiles : le nom de la tâche, le type de tâche, la coordonnée X.

Pour la seconde solution, le traitement se faisant en parallèle au traitement de la première solution, nous passons sur chaque noeud et prenons le noeud avec la coordonnée X la plus petite. Nous faisons le même traitement pour chaque sous-noeud d'un noeud.

Grâce à cela, nous avons notre nouveau fichier XML avec toutes les données nécessaires triées.

Maintenant, la question est : que fait-on avec ce fichier ?

Nous répondons au besoin cité précédemment. Nous générons une formation générique qui aurait pour objectif d'être donnée aux pilotes. Pour cela, nous reprenons dans ce fichier chaque tâche correspondant à une tâche utilisateur [2] tâche abstraite utilisateur, tâche moteur, tâche cognitive, etc. et nous les plaçons, surlignées, dans un fichier pdf.

Nous avons comme résultat pour cette application une formation générique par rapport au modèle de tâches de chaque prototype/interface.

3 Présentation

La figure 4.1 montre l'écran sur lequel nous arrivons lorsque nous lançons l'application. Cet écran permet plusieurs choses. Premièrement, il permet de choisir un fichier (voir figure 4.2) que celui-ci soit un fichier venant de Hamsters (.hmst) ou un fichier .xml (voir figure 4.3). Deuxièmement, l'application permet d'afficher le PDF que l'on vient de générer avec l'application (voir figure 4.4).

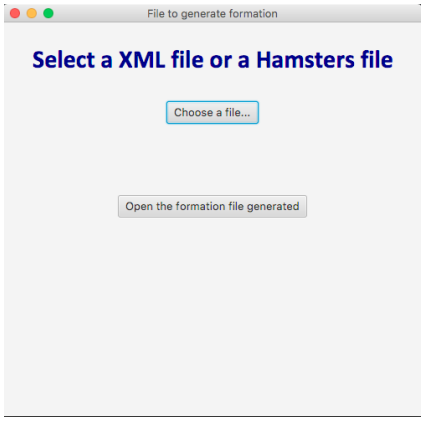


FIGURE 4.1 – Écran d'accueil

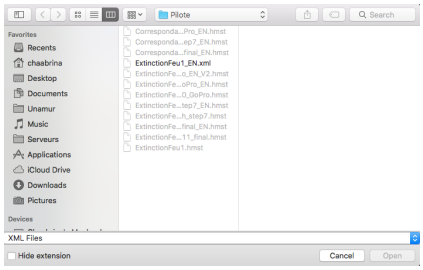


FIGURE 4.2 – Écran pour le choix d'un fichier

2. Ces tâches sont décrites à la figure 2.16 de la section 5.3.3 de la partie "état de l'art".

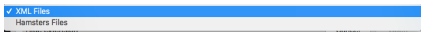


FIGURE 4.3 – Écran montrant la possibilité de choisir un format Hamsters ou un format XML

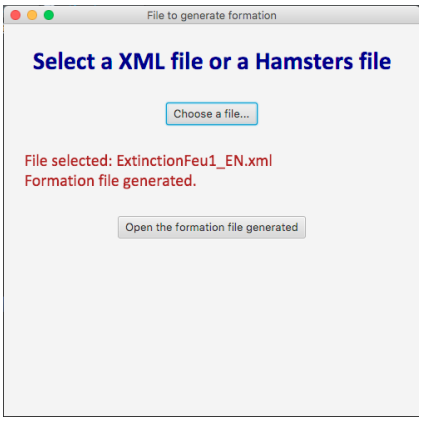


FIGURE 4.4 – Écran montrant quel fichier a été sélectionné et montrant que le fichier de formation a été généré

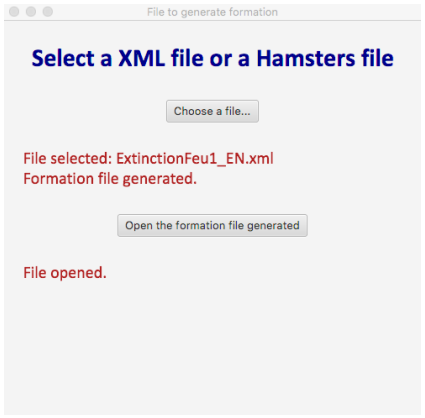


FIGURE 4.5 – Écran montrant que le PDF a été ouvert

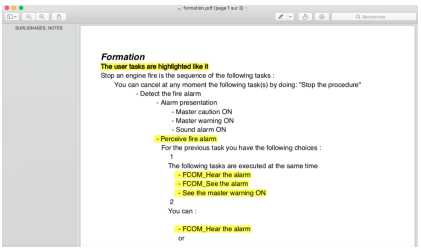


FIGURE 4.6 – Écran montrant le PDF

Enfin, nous avons un PDF qui est généré sur base du fichier fourni. Celui-ci fournit toutes les tâches utilisateurs quelles qu'elles soient, c'est-à-dire moteur, cognitive, abstraite, etc. mais également les autres tâches. Pour différencier les différents types de tâches, utilisateurs ou non, les tâches utilisateurs ont été surlignées dans le document. Nous parlerons de la validation de ce document dans la section 1 du chapitre 5.

Quatrième partie

Conclusion et travaux futurs

Chapitre 5

Conclusion

Sommaire

1 Améliorations possibles et travaux futurs

Dans cette section, des améliorations sont présentées ainsi que des possibilités de recherches sur la thématique de ce travail.

Comme l'a dit le navigateur français Bruno Peyron,

"On peut toujours améliorer un record. On peut toujours faire mieux."

Le travail fait ici peut donc être forcément amélioré. Pour l'application réalisée de génération de formation, plusieurs améliorations sont possibles.

Dans un premier temps, l'interface graphique pourrait être rendue plus agréable à utiliser en la rendant plus attrayante avec des couleurs, une police d'écriture dite "agréable à lire". Ce type de changement peut être fait en menant une enquête auprès d'utilisateurs potentiels de l'application.

Dans un deuxième temps, on peut imaginer de créer plus de filtres pour la personne qui génère. Par exemple, on permettrait à l'utilisateur de mettre un filtre pour ne garder que les tâches abstraites afin d'avoir une vue d'ensemble de la procédure.

Dans un troisième temps, une amélioration du traitement du fichier peut être effectuée. Par exemple, on pourrait clairement imaginer avoir une formation sous forme de texte continu plutôt que de liste à puces.

Dans un dernier temps, une validation de cette application et du fichier généré serait idéale. Pour cela, nous pouvons imaginer réunir un panel de pilotes auxquels nous donnerions notre fichier de formation. Ensuite, nous les mettrions en face d'une des interfaces et nous verrions à quel point ces pilotes s'en sortent. D'autres améliorations sont surement possibles celles citées ci-dessus ne sont pas exhaustives.

Une piste de futur projet serait d'utiliser le document XML fourni par Hamsters, dans notre cas, pour chaque modèle de tâches de chaque prototype. Avec chaque modèle de tâches qui ne sont finalement que des arbres XML, on pourrait montrer à quel point ils sont similaires grâce à la structure d'arbre qu'ils possèdent ainsi qu'à la technique du calcul de distance entre les arbres.

2 Discussion et Conclusion

Les cockpits interactifs ont pour objectif dans un futur proche de remplacer le cockpit physique actuel. Le cockpit actuel est composé de énormément de boutons avec lesquels les pilotes interagissent avec les sensations de toucher. Ceux-ci peuvent reconnaitre un bouton grâce à la taille des crans, au nombre de crans, etc.

De toute évidence, avec une interface tactile, il est beaucoup moins aisé de pouvoir ressentir quel bouton est manipulé grâce aux crans ou toutes autres compositions physiques. Cependant, en réponse à la question que nous nous sommes posée plus tôt, il est possible d'imaginer qu'un bouton aurait sa propre séquence de vibrations. La vibration pourrait varier selon le bouton de par son intensité, de par son occurrence (le bouton X vibrerait deux fois au toucher alors que le bouton Y vibrerait quatre fois rapidement), etc.

Encore une fois, avec une interaction tactile, les questions de sécurité données dans l'introduction sont difficiles à aborder. En effet, dans un cockpit, à l'heure actuelle, plusieurs niveaux de sécurité sont mis en place. Pour notre procédure d'extinction d'un feu moteur, on a plusieurs niveaux de sécurité. Tout d'abord le pilote doit retirer le capot du bouton "Engine fire" avec une certaine force et jusqu'à son maximum sinon celui-ci se referme automatiquement. Ensuite, le pilote doit appuyé sur ce même bouton "Engine fire" en restant légèrement appuyé, celui-ci ressort une fois relâché. Enfin, il doit appuyer sur le bouton "agent" afin de le vider.

Pour notre interaction tactile, nous avons essayé de rester dans un même niveau de granularité avec de respecter les niveaux de sécurité. Effectivement, par rapport au modèle de tâches, nous sommes descendu au même niveau de granularité afin de respecter la similarité au maximum. La série d'actions reste sensiblement la même : le pilote prend la protection, la relève à son maximum (elle redescend si ce n'est pas le cas, comme pour le physique) et il appuie sur le bouton "engine fire" de manière prolongée afin d'armer les agents. Une action prolongée qui se termine par une vibration puisque nous ne pouvions avoir le ressenti d'un ressort. Enfin, il appuie sur l'agent comme pour l'interaction physique. En gardant la même série d'actions et le même niveau de granularité pour l'interaction tactile, nous minimisons au maximum le nombre d'erreurs possibles et nous le mettons au maximum au même niveau que pour l'interaction physique.

Dans le travail, nous avons parlé de niveau de granularité. Dans le cas de notre étude, nous savons que nous avons atteint le niveau de granularité nécessaire puisque nous savons comparer l'ensemble des modèles de tâches. Cependant, selon l'étude ou la recherche que nous souhaitons mené, le niveau de granularité peut être plus élevé pour les besoins ou moins élevé. Le niveau de granularité nécessaire ou non est sujet à la discussion.

Les avantages de faire entrer le tactile dans le cockpit afin de le transformer en cockpit interactif sont multiples. D'une part, le tactile amené dans le cockpit réduira probablement les coûts de production. En effet, dans un cockpit, le panneau plafond actuel coûte une véritable fortune. Le fait d'introduire le tactile amènera moins de coûts quant aux composants et ne demandera que le coût de l'installation tactile. D'autre part, la technologie tactile est connue de tous. Elle

permet aux pilotes de nouvelles générations d'appréhender le cockpit de manière plus simple, plus rapide et plus efficace.

Enfin, les modèles de tâches et le prototypage combinés permettent de développer une interaction similaire à une interaction physique existante. Ce travail amène quelques chemins de réflexions et moyens de conceptions afin de développer des interactions tactiles dans le cadre de l'aviation civile tout en respectant les procédures déjà existantes. Il a contribué à la sortie d'un article scientifique qui se trouve en annexe C.

Bibliographie

[Bal94] Sandrine Balbo. *Evaluation ergonomique des interfaces utilisateur : un pas vers l'automatisation.* PhD thesis, Université Joseph-Fourier-Grenoble I, 1994.

[BL97] M. Beaudouin-Lafon. Ingénierie des systèmes interactifs. http ://www-ihm.lri.fr/ mbl/ENS/IHM/ecole-in2p3/Cours/index.html, 1997.

[Bla11] Renaud Blanch. Modéles de tâches. 2011.

[CNS+95] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M Young. Four easy pieces for assessing the usability of multimodal interaction : the care properties. In *Human—Computer Interaction*, pages 115–120. Springer, 1995.

[Cro17] Martin Cronel. *Une approche pour l'ingénierie des systèmes interactifs critiques multimodaux et multi-utilisateurs : Application à la prochaine génération de cockpit d'aéronefs.* PhD thesis, Université de Toulouse 3 Paul Sabatier, 2017.

[CSS08] Sybille Caffiau, Dominique L Scapin, and Loé Sanou. Retour d'expérience en enseignement de la modélisation de tâches. *ERGO IA 2008*, 2008.

[FJ14] Laurence Nigay Frédéric Jourde, Yann Laurillau. Description des tâches avec un système interactif multiutilisateur et multimodal : Etude comparative de notations. Journal d'Interaction Personne-Système (JIPS), AFIHM, 2014.

[GCS] Patrick Girard, Sybille Caffiau, and Dominique Scapin. K-made, un outil de modélisation des tâches pour l'enseignement et la recherche.

[GGSL92] Jérôme Gensel, Pierre Girard, and Projet SHERPA—IRIMAG-LIFIA. Expression d'un modéle de tâches á l'aide d'une représentation par objets. In *Actes de la Conférence Représentations Par Objets (RPO'92), La Grande Motte, France*, pages 225–236, 1992.

[GUI11] Romaric GUILLERM. *Intégration de la sûreté de fonctionnement dans les processus d'ingénierie système.* PhD thesis, Université de Toulouse III - Paul Sabatier, 2011.

[HAR16] Frédéric HARDY. Do 178, le développement de logiciels critiques, 2016.

[HH] W. BO H. HRISTOV. Safety critical computer systems : Failure independence and software diversity effects on reliability of dual channel structures.

[Ily] CICEKLI Ilyas. Similarities and differences.

[IRI] IRIT. https ://www.irit.fr/recherches/ics/softwares/hamsters/types.html.

[I.S] I.S.I.T. Do-178 : Répondre aux exigences de la norme de sûreté de fonctionnement logiciel avionique.

[JC] Alexandre DEMEURE et Lionel BALME Joëlle COUTAZ, Gaëlle CALVARY. *Systèmes interactifs et adaptation centrée utilisateur : la plasticité des Interfaces Homme-Machine*, chapter Chapitre 9. Hermes Sciences Publishing Ltd.

[LAP] Jean-Claude LAPRIE. Sûreté de fonctionnement des systèmes : concepts de base et terminologie.

[LCG89] Jean-Claude Laprie, Bernard Courtois, and Marie-Claude Gaudel. Sûreté de fonctionnement des systèmes informatiques. 1989.

[NPHDP] David NAVARRE, Philippe PALANQUE, Arnaud HAMON, and Sabrina DELLA PASQUA. Similarity as a design driver for user interfaces of dependable critical systems.

[Pat04] Fabio Paternó. An engineered notation for task models. 2004.

[PB] Ph. Palanque and R. Bastide. A formalism for reliable user interfaces.

[PL03] TACTUAL PERCEPTION and Jack M. Loomis. Chapter 31 tactual perception. 2003.

[Pot12] Frederic Pothon. Do-178c/ed-12c versus do-178b/ed-12b : Changes and improvements. *ACG Solutions*, 2012.

[RCP14] Fahssi Racim, Martinie Célia, and Palanque Philippe. Hamsters : un environnement d'édition et de simulation de modèles de tâches. In *IHM'14, 26e conférence francophone sur l'Interaction Homme-Machine*, pages 5–6, 2014.

[RGC11] Daniel Read and Yael Grushka-Cockayne. The similarity heuristic. *Journal of Behavioral Decision Making*, 24(1) :23–46, 2011.

[RMWZ14] Martin P. Robillard, Walid Maalej, Robert J. Walker, and Thomas Zimmermann, editors. *Recommendation Systems in Software Engineering*, chapter Basic Approaches in Recommendation Systems, pages 15–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[RTC12] Rtca : Do-178c / ed-12c : Software considerations in airborne systems and equipment certification, 2012.

[SGGC08] Loé Sanou, Patrick Girard, Laurent Guittet, and Sybille Caffiau. Tester la conformité d'une ihm á son modéle de tâches. In *Proceedings of the 20th Conference on l'Interaction Homme-Machine*, pages 159–162. ACM, 2008.

[Sé06] SORLIN Sébastien. *Mesurer la similarité de graphe*. PhD thesis, Université Claude Bernard Lyon I, 2006.

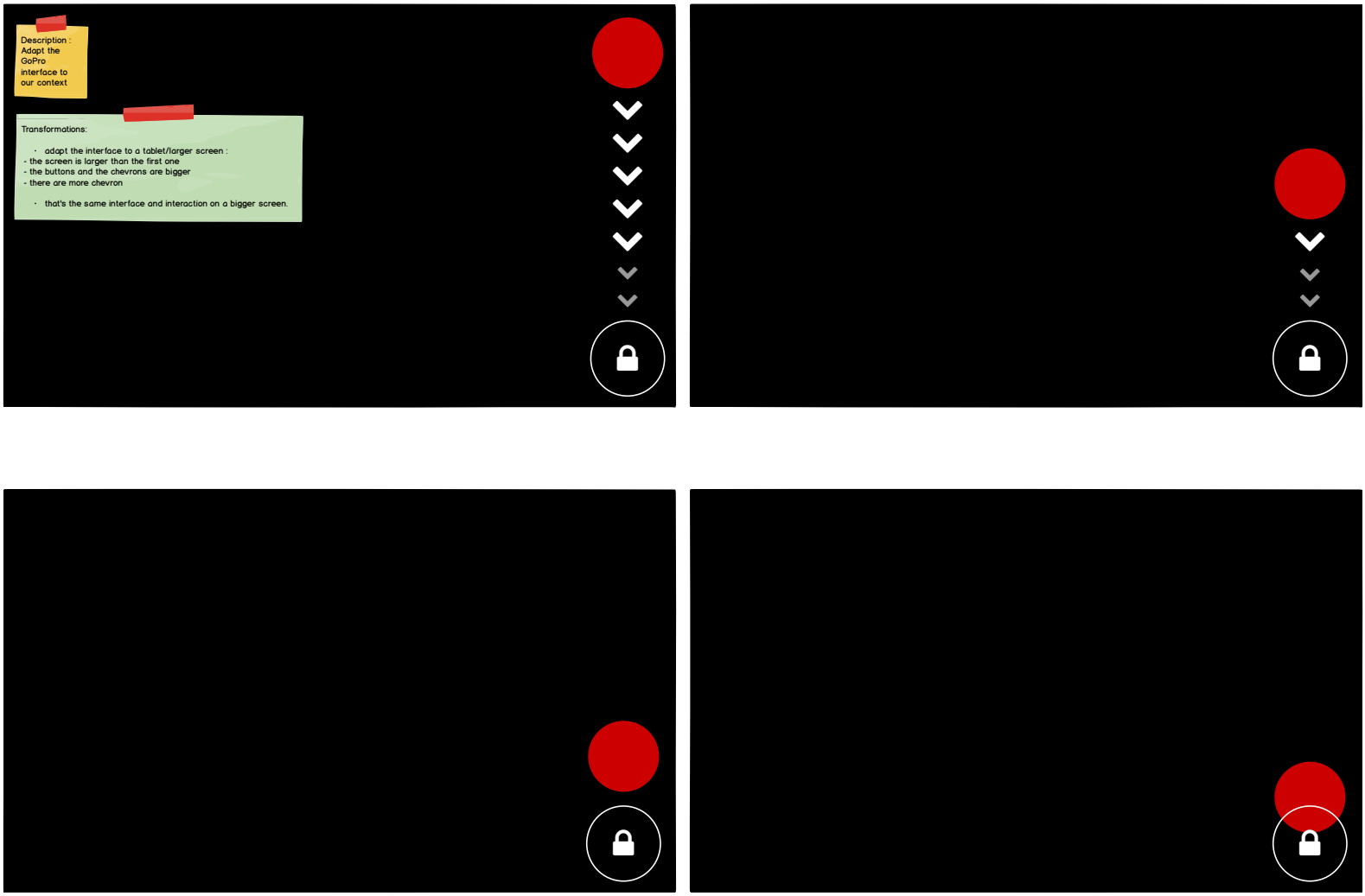
[Tor] Bertrand Tornil. Adaptations et interactions gestuelles et haptiques, ciblées utilisateurs.vers plus d'utilisabilité et d'accessibilité.

[Ver] Verifysoft. La norme do-178c.

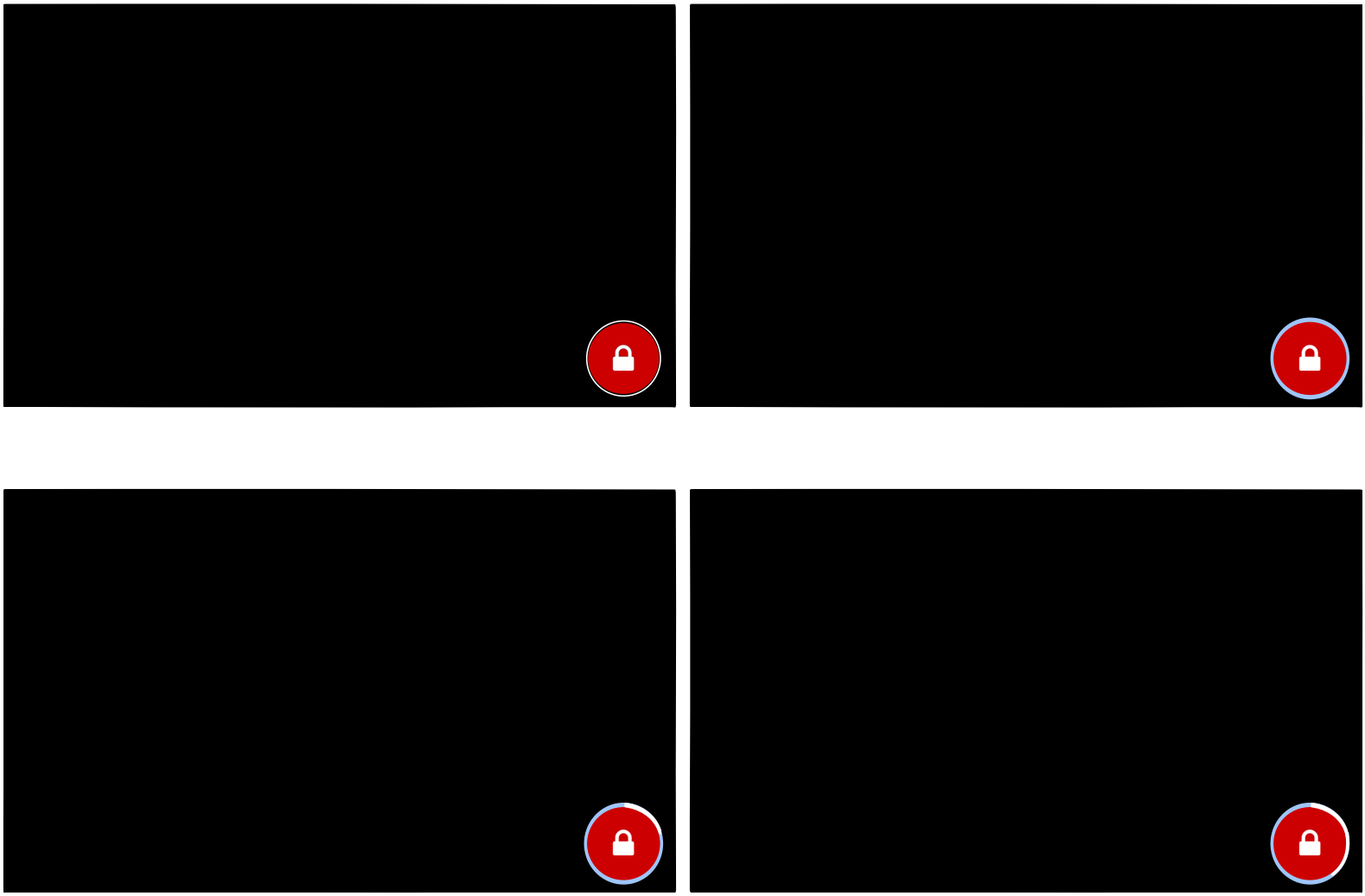
Annexe A

Prototypes

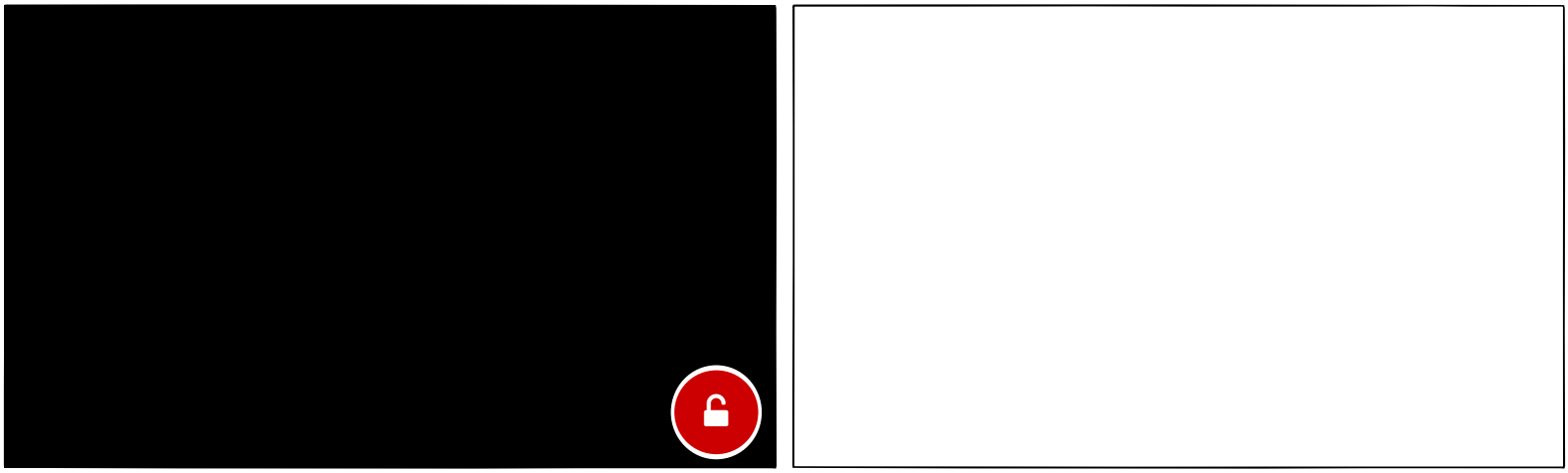
Prototype 1


Description :
Adapt the
GoPro
interface to
our context
Transformations:
· adapt the interface to a tablet/larger screen :
- the screen is larger than the first one
- the buttons and the chevrons are bigger
- there are more chevron
· that's the same interface and interaction on a bigger screen.

Prototype 1



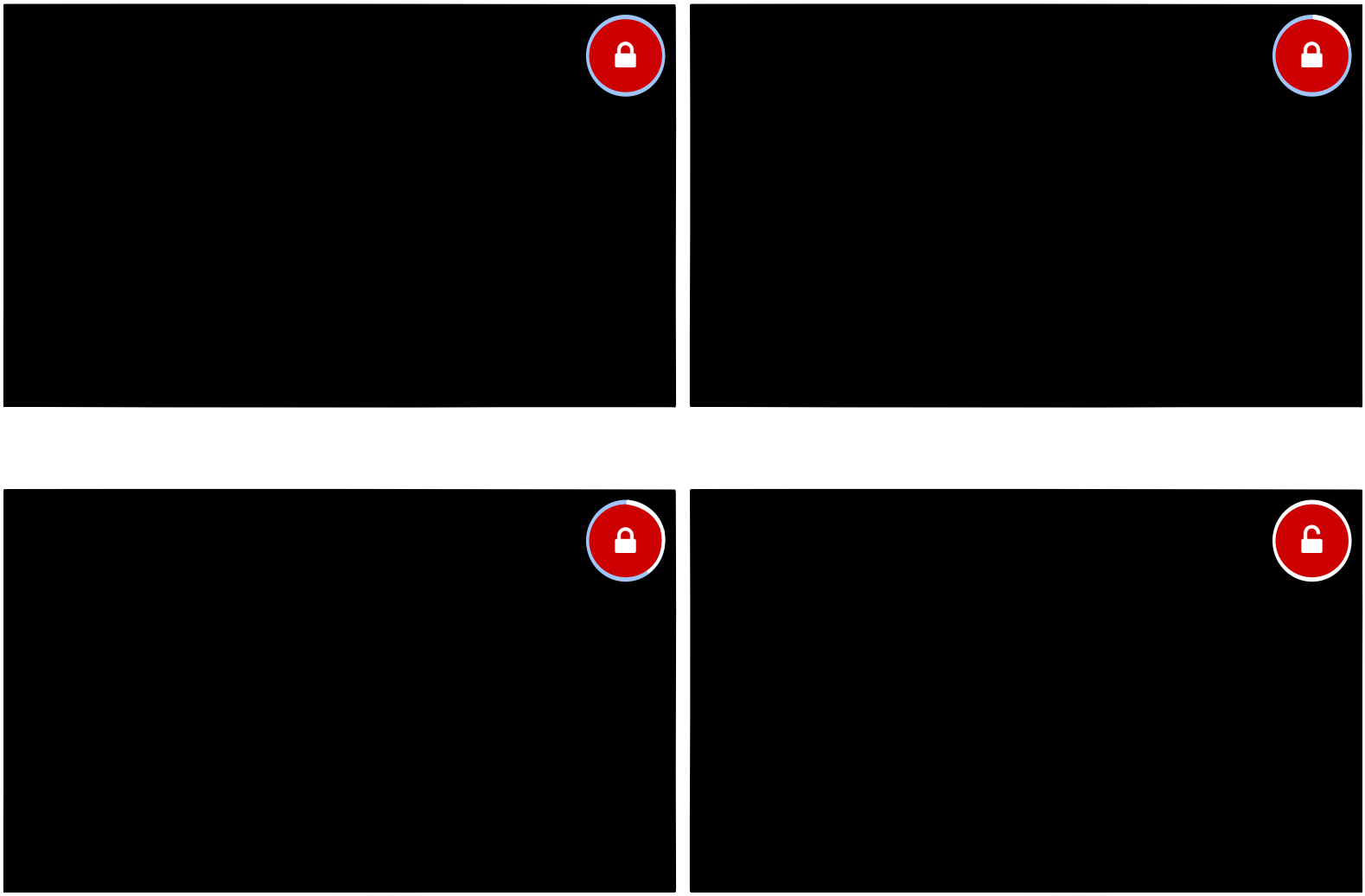
Prototype 1



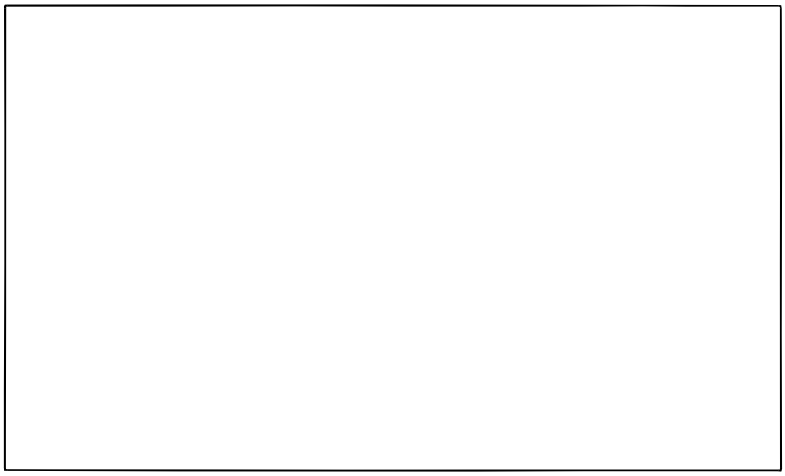
Prototype 2


Description : Change the way to interact with the button
Transformations:
the two buttons are switched
the interaction with the button is now from the bottom of the screen to the top of the screen

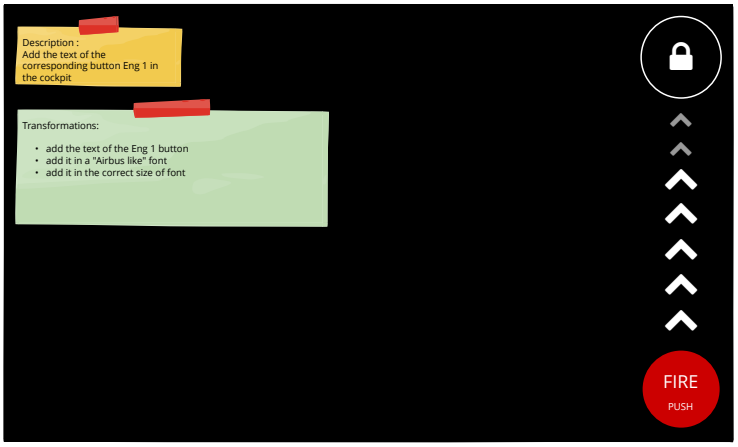
Prototype 2



Prototype 2



Prototype 3


Description :
Add the text of the corresponding button Eng 1 in the cockpit
Transformations:
• add the text of the Eng 1 button
• add it in a "Airbus like" font
• add it in the correct size of font
FIRE
PUSH

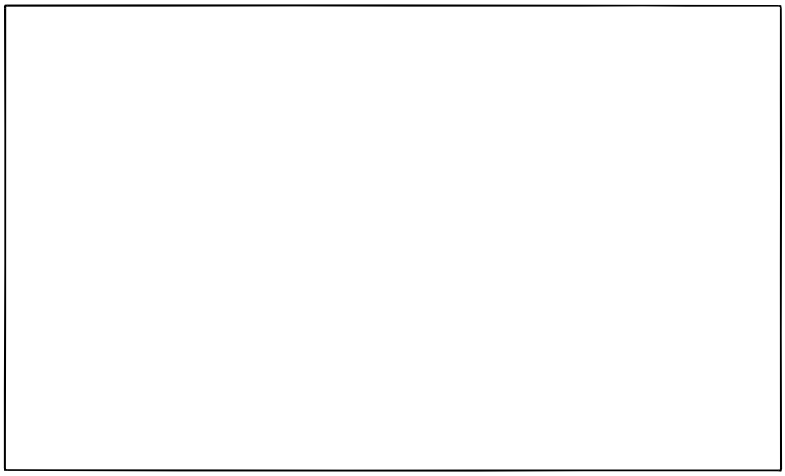

FIRE
PUSH


FIRE
PUSH


FIRE
PUSH

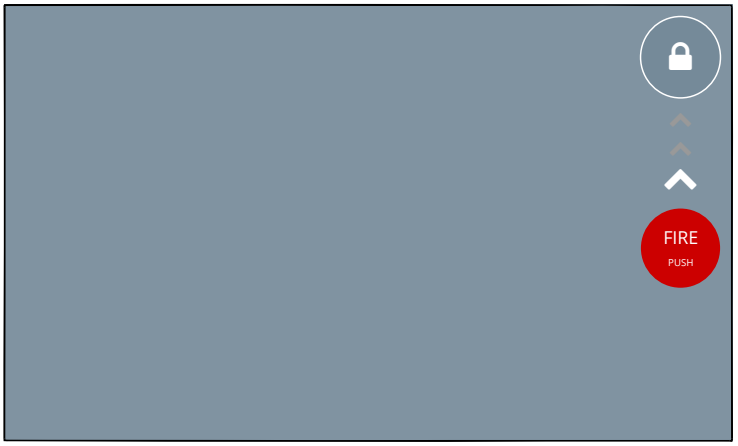
Prototype 3

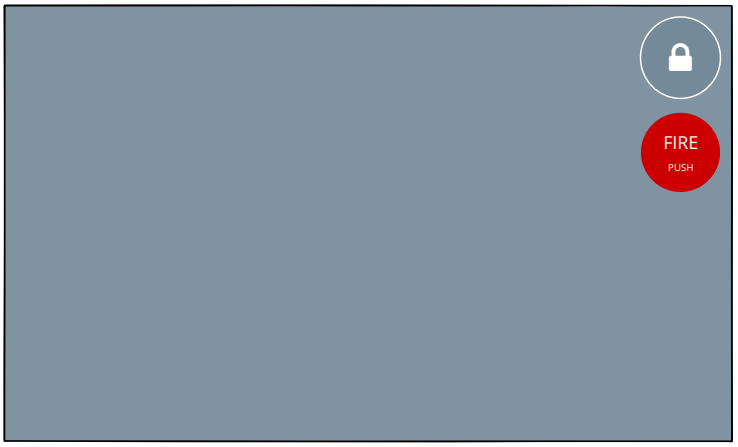
Prototype 3

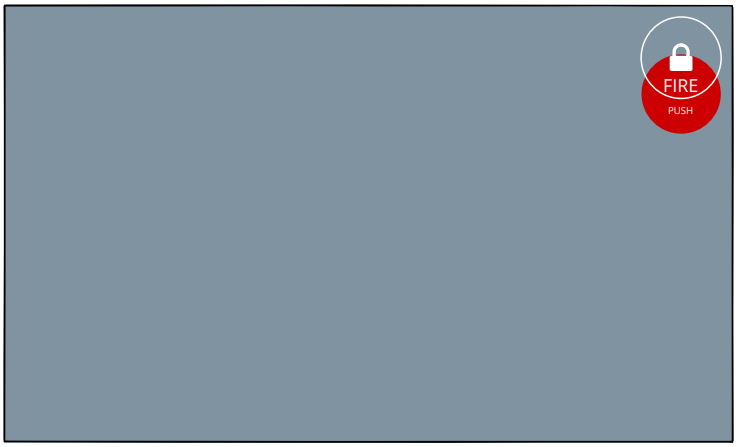


Prototype 4

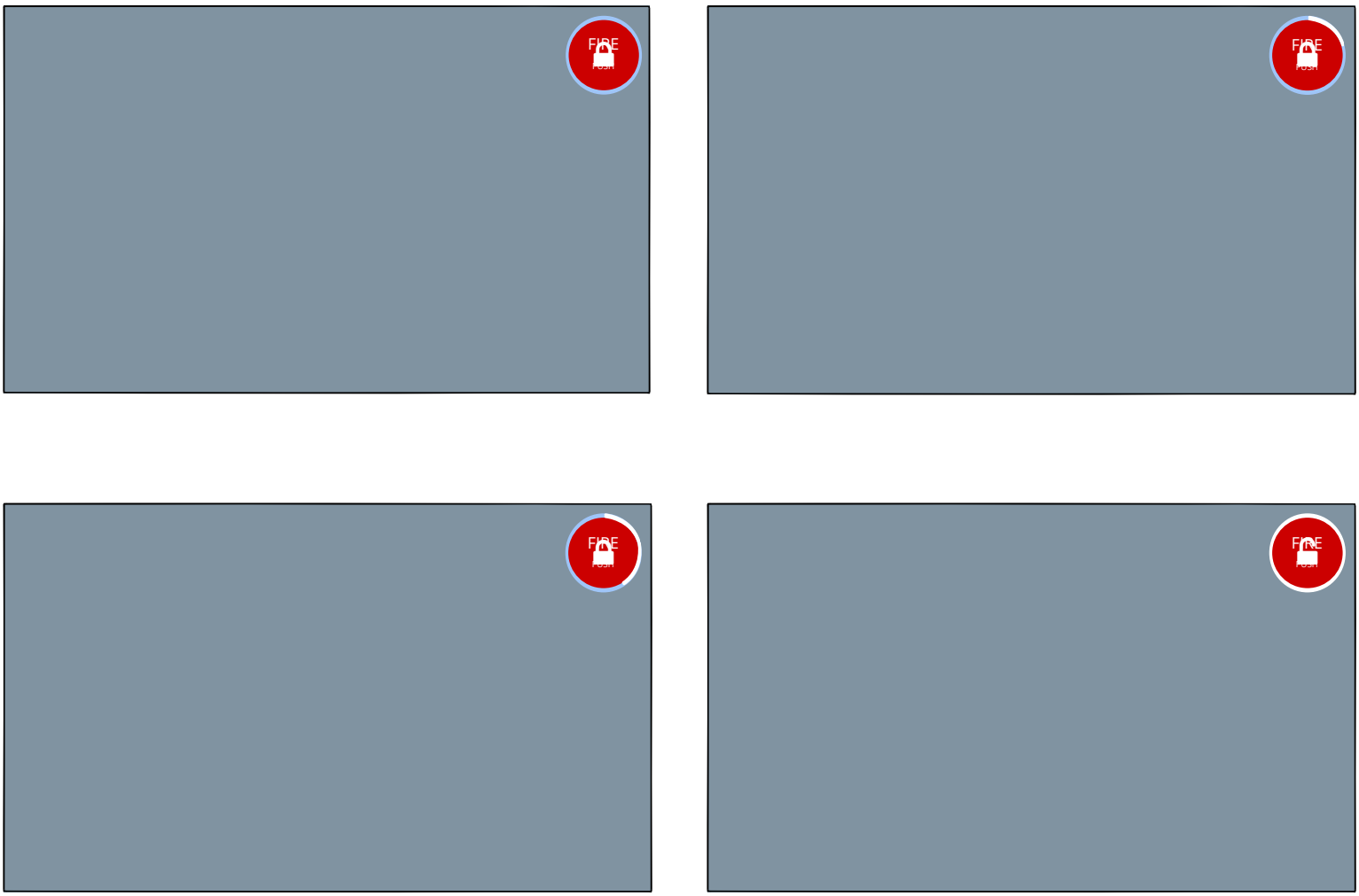

Description :
Change the color of the background corresponding to the background color of the head panel
Transformations:
• change the background color in a "Airbus like" background color
FIRE
PUSH


FIRE
PUSH

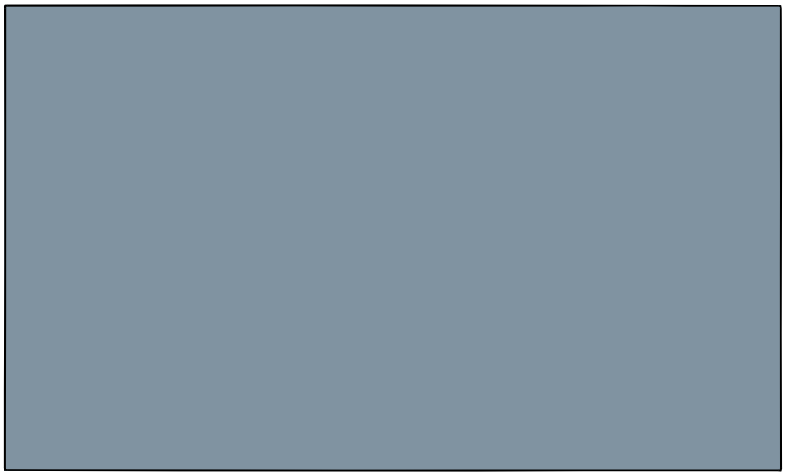

FIRE
PUSH


FIRE
PUSH

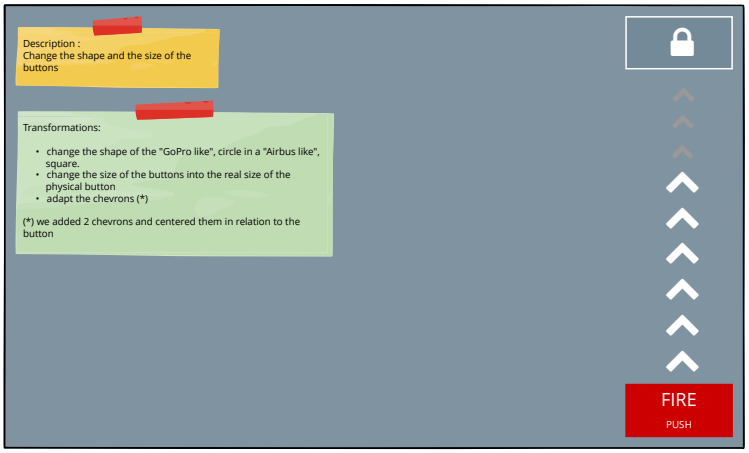
Prototype 4

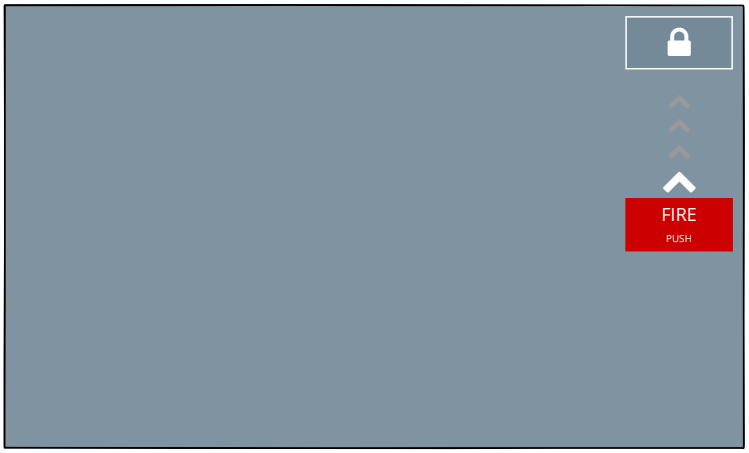


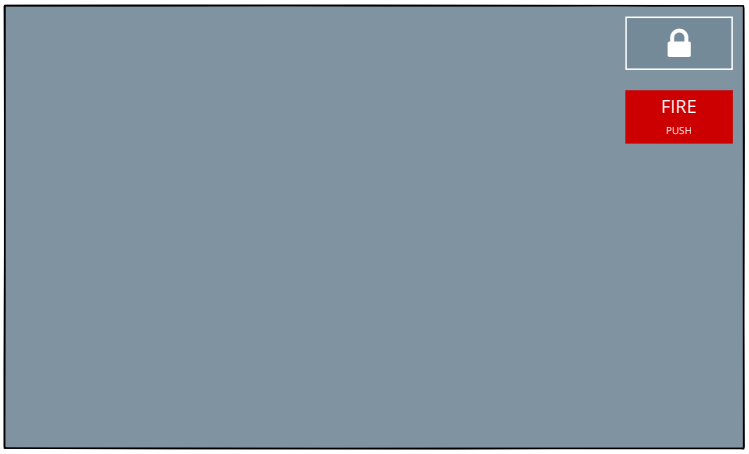
Prototype 4

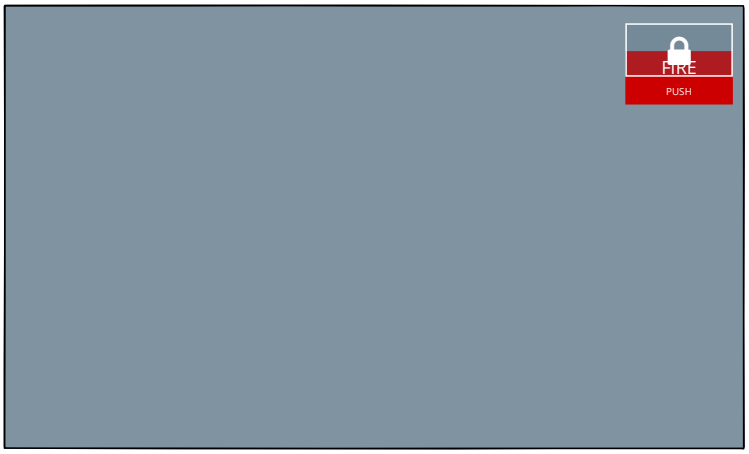


Prototype 5

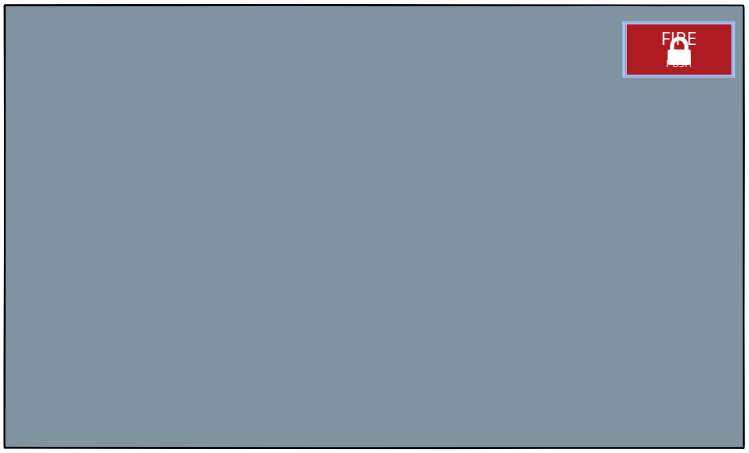

Description :
Change the shape and the size of the buttons
Transformations:
• change the shape of the "GoPro like", circle in a "Airbus like", square.
• change the size of the buttons into the real size of the physical button
• adapt the chevrons (*)
(*) we added 2 chevrons and centered them in relation to the button
FIRE
PUSH

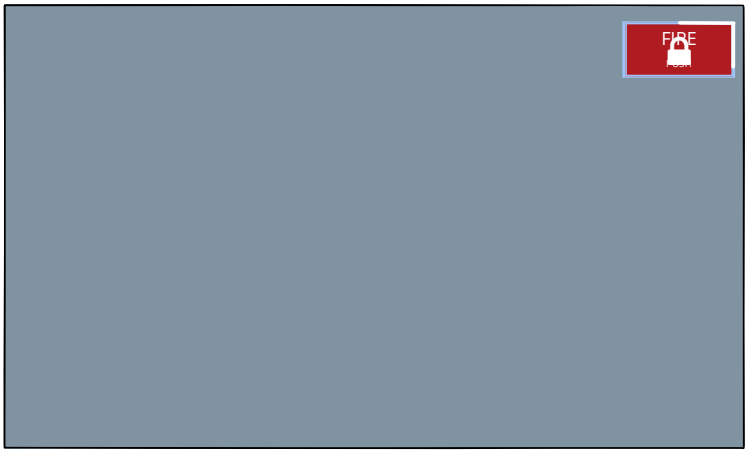

FIRE
PUSH

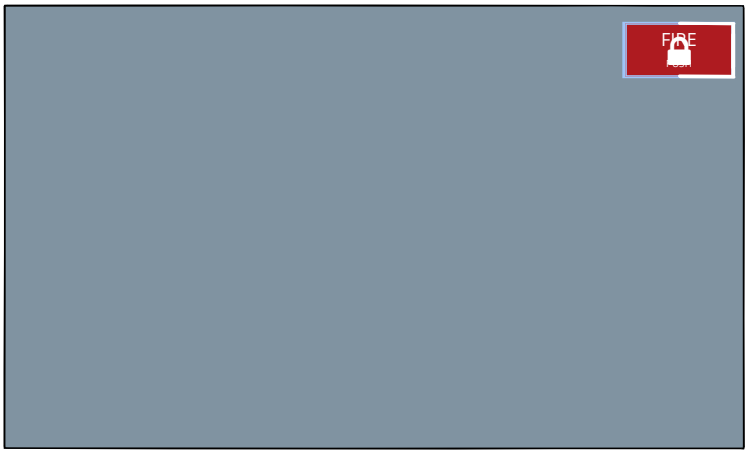

FIRE
PUSH

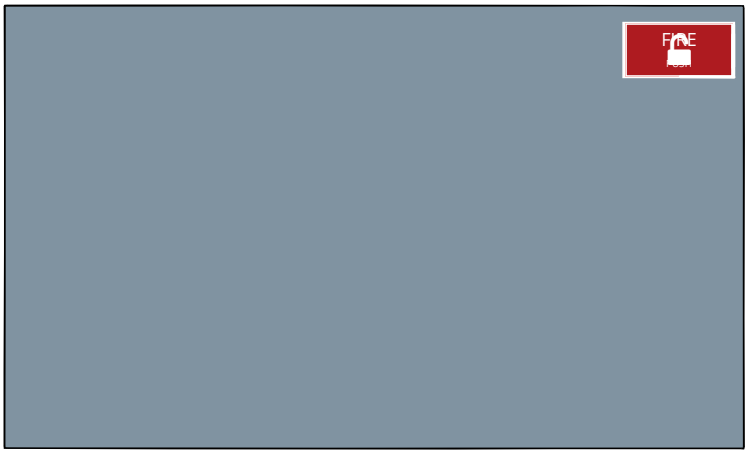

FIRE
PUSH

Prototype 5

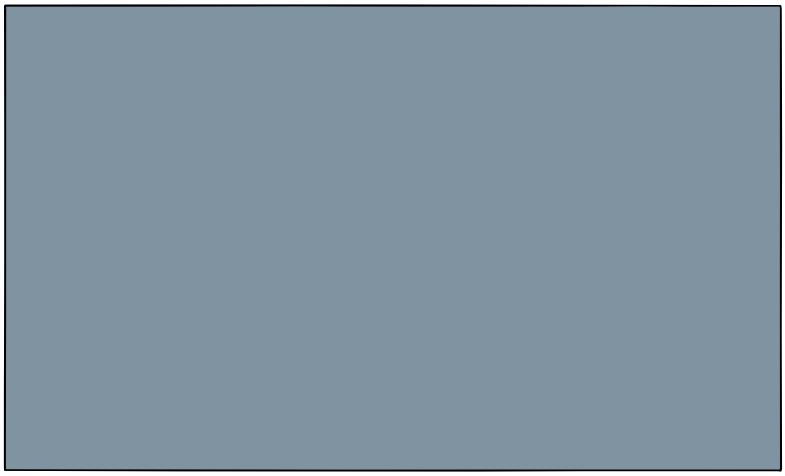




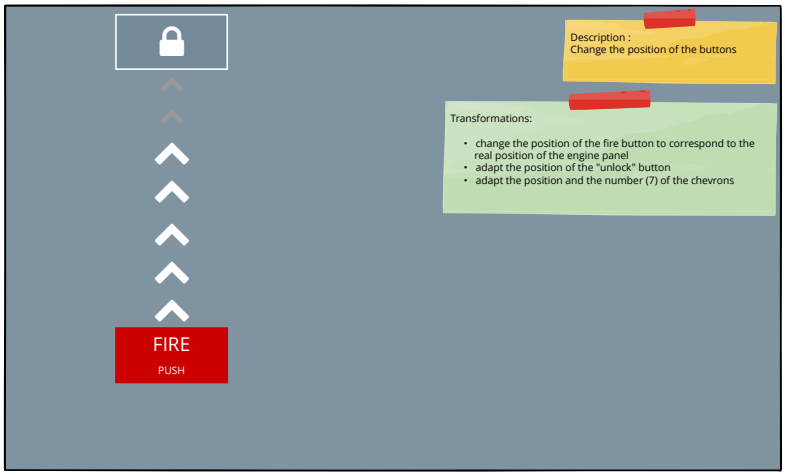


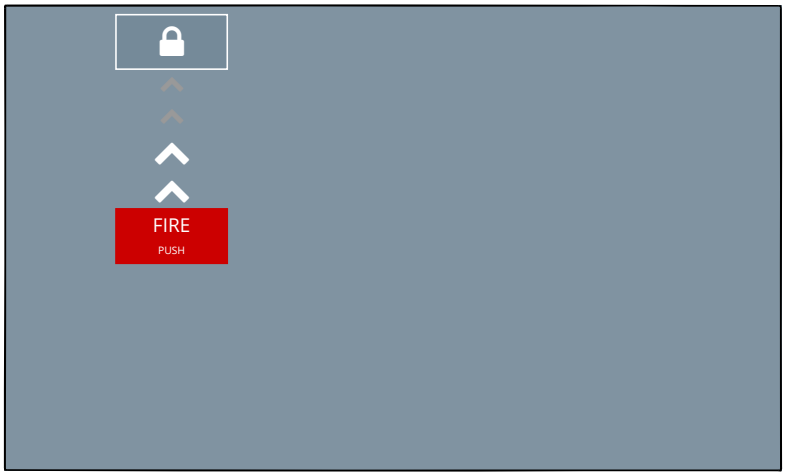


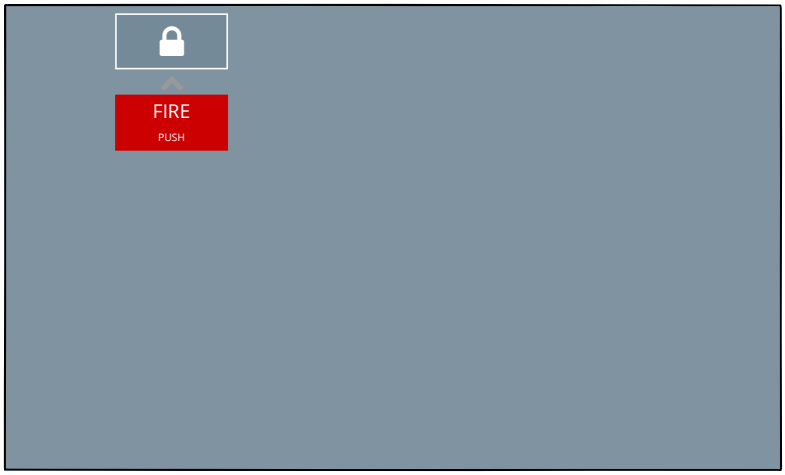
Prototype 5

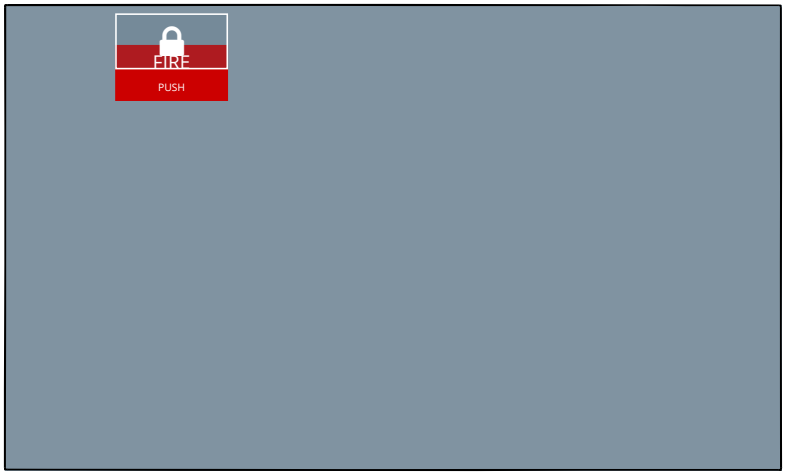


Prototype 6

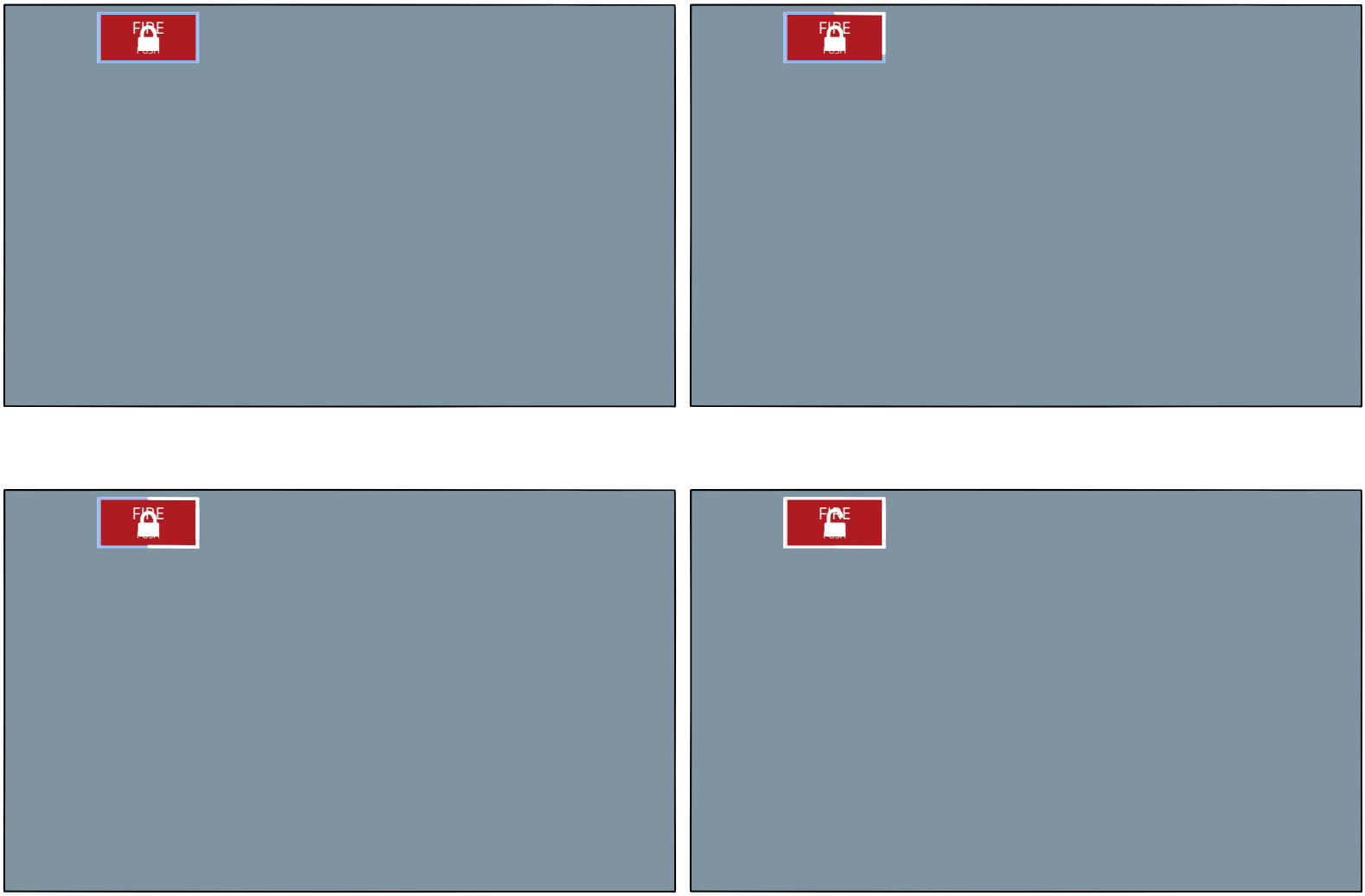

Description :
Change the position of the buttons
Transformations:
• change the position of the fire button to correspond to the real position of the engine panel
• adapt the position of the "unlock" button
• adapt the position and the number (7) of the chevrons
FIRE
PUSH


FIRE
PUSH

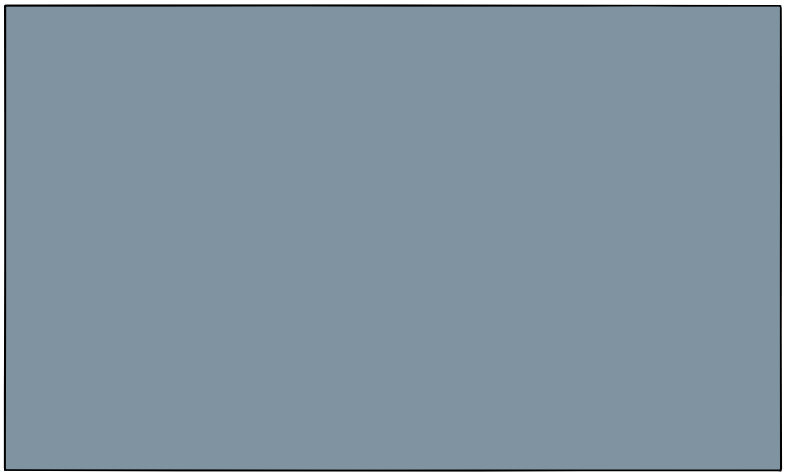

FIRE
PUSH


FIRE
PUSH

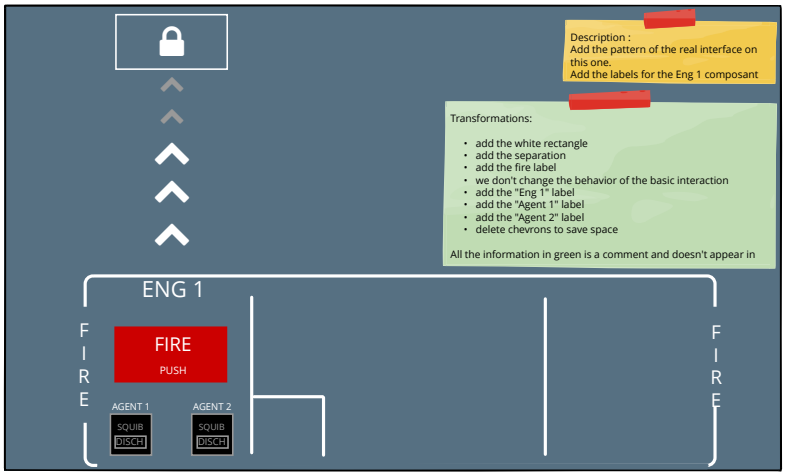
Prototype 6

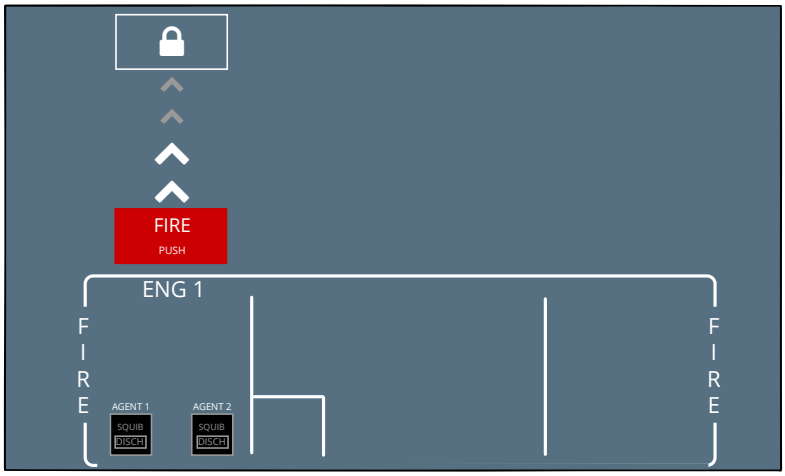


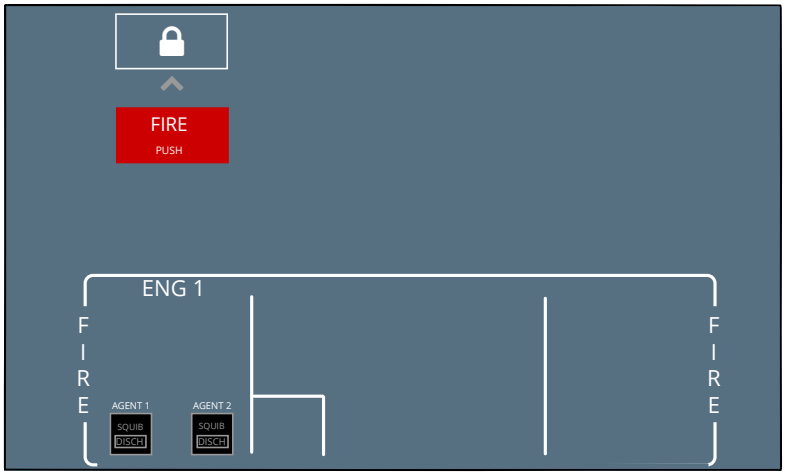
Prototype 6

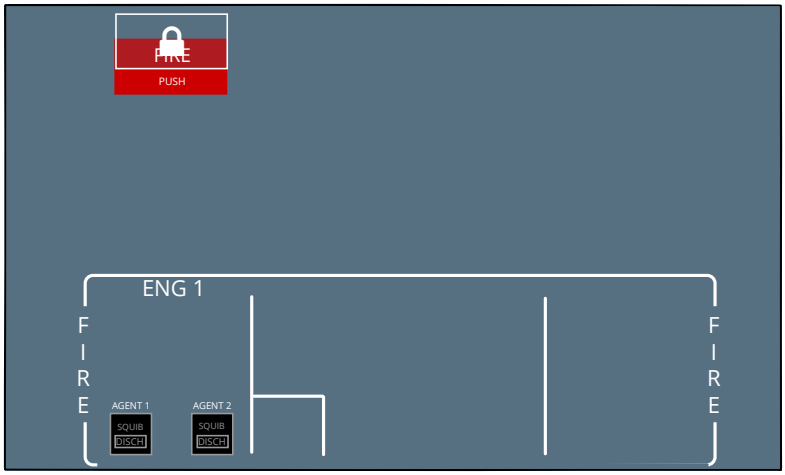


Prototype 8

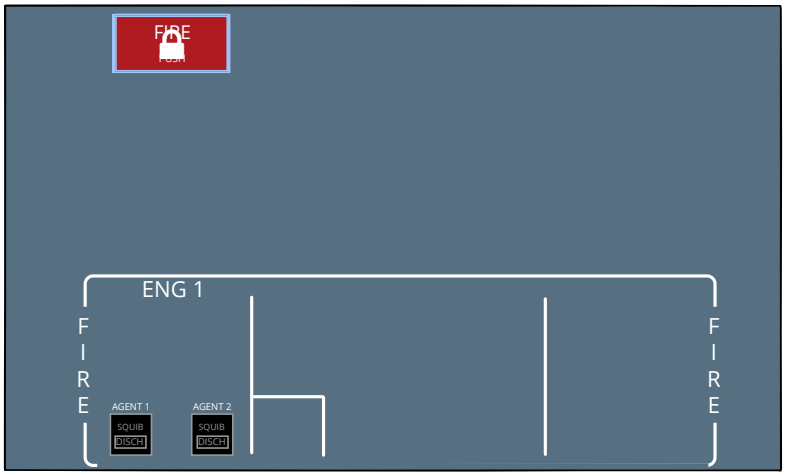

Description :
Add the pattern of the real interface on this one.
Add the labels for the Eng 1 composant
Transformations:
add the white rectangle
add the separation
add the fire label
we don't change the behavior of the basic interaction
add the "Eng 1" label
add the "Agent 1" label
add the "Agent 2" label
delete chevrons to save space
All the information in green is a comment and doesn't appear in
ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

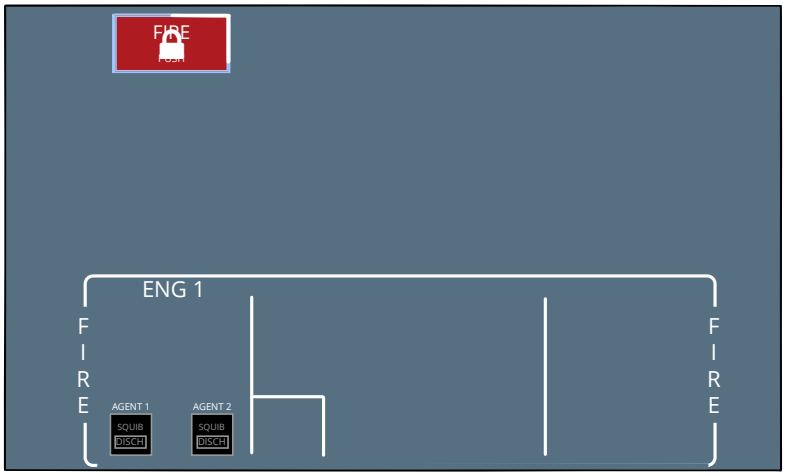

FIRE
PUSH
ENG 1
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

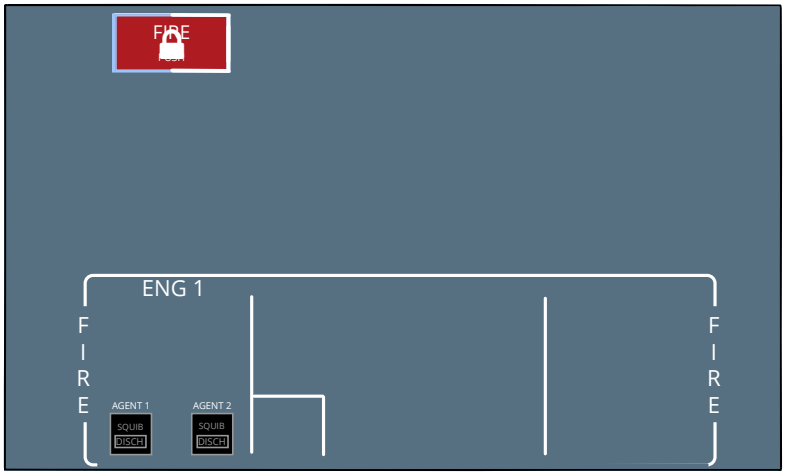

FIRE
PUSH
ENG 1
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

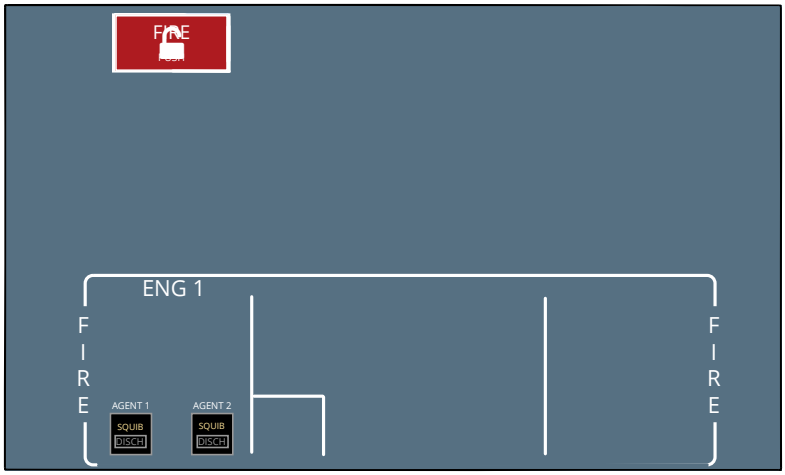

FIRE
PUSH
ENG 1
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

Prototype 8

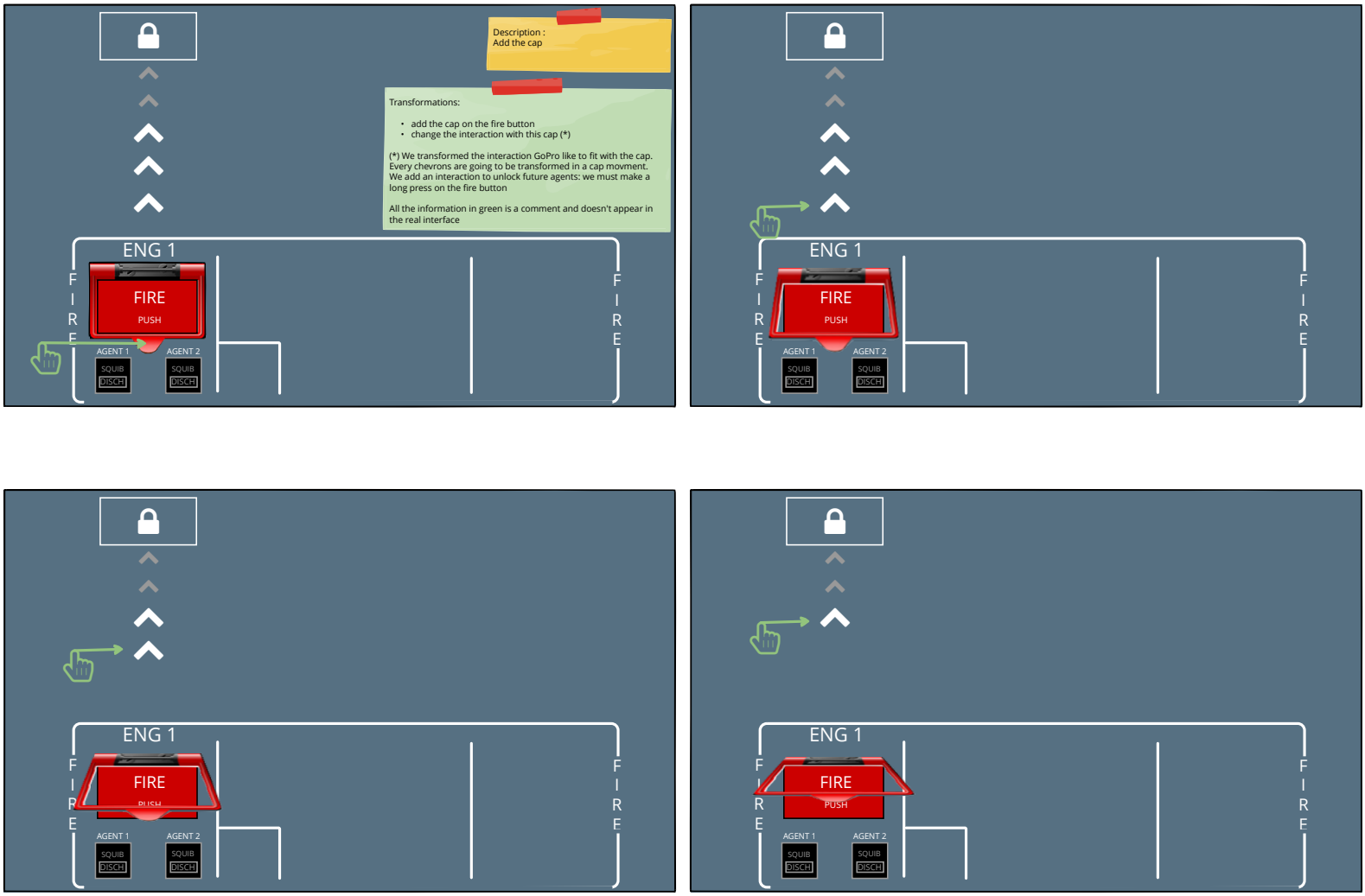

FIRE
ENG 1
FIRE
AGENT 1
SQUIB
DISCH
AGENT 2
SQUIB
DISCH
FIRE


FIRE
ENG 1
FIRE
AGENT 1
SQUIB
DISCH
AGENT 2
SQUIB
DISCH
FIRE

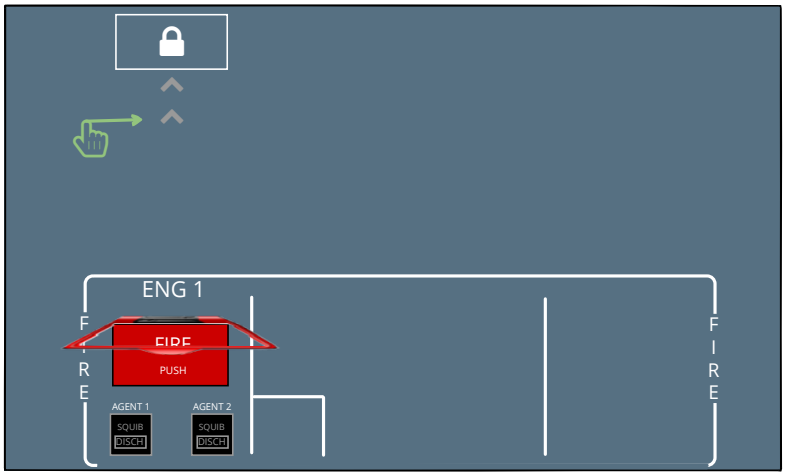

FIRE
ENG 1
FIRE
AGENT 1
SQUIB
DISCH
AGENT 2
SQUIB
DISCH
FIRE

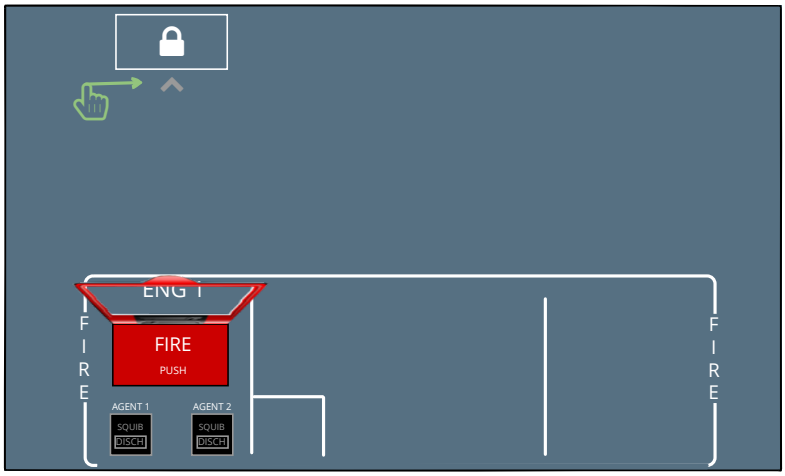

FIRE
ENG 1
FIRE
AGENT 1
SQUIB
DISCH
AGENT 2
SQUIB
DISCH
FIRE

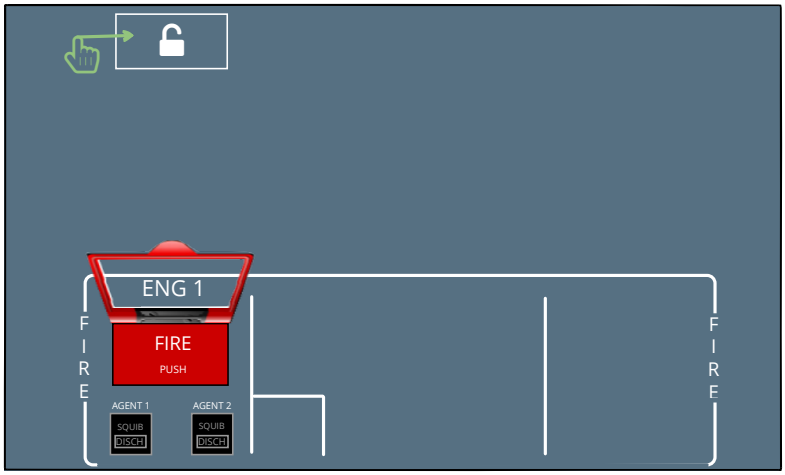
Prototype 9

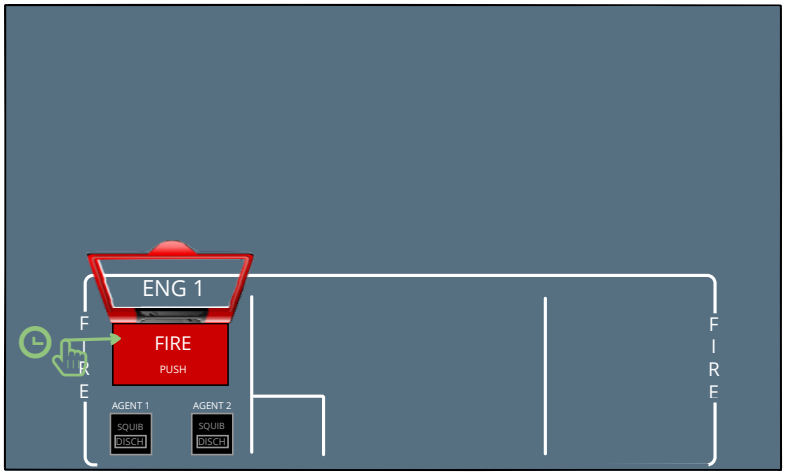

Description :
Add the cap
Transformations:
add the cap on the fire button
change the interaction with this cap (*)
(*) We transformed the interaction GoPro like to fit with the cap. Every chevrons are going to be transformed in a cap movment. We add an interaction to unlock future agents: we must make a long press on the fire button
All the information in green is a comment and doesn't appear in the real interface
ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE

Prototype 9

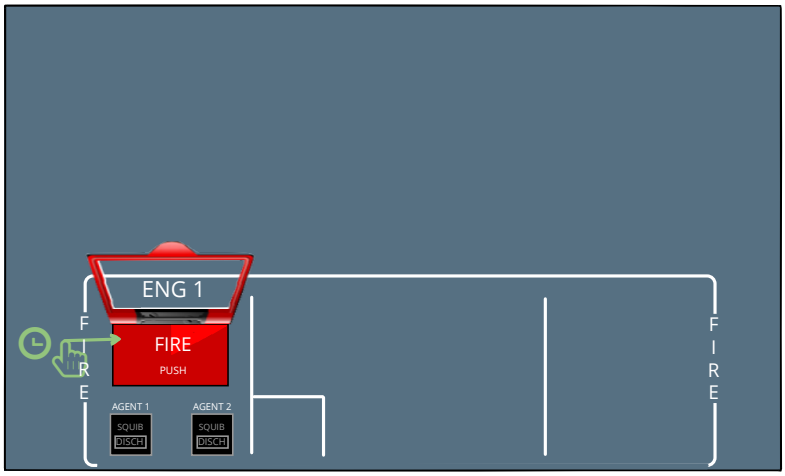

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

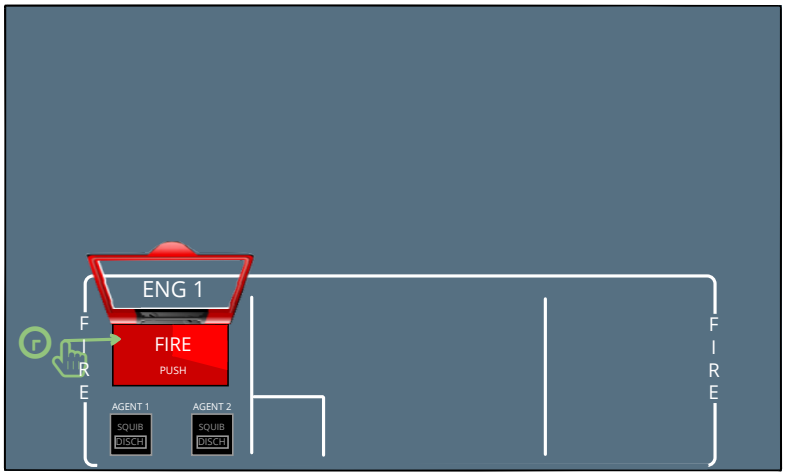

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

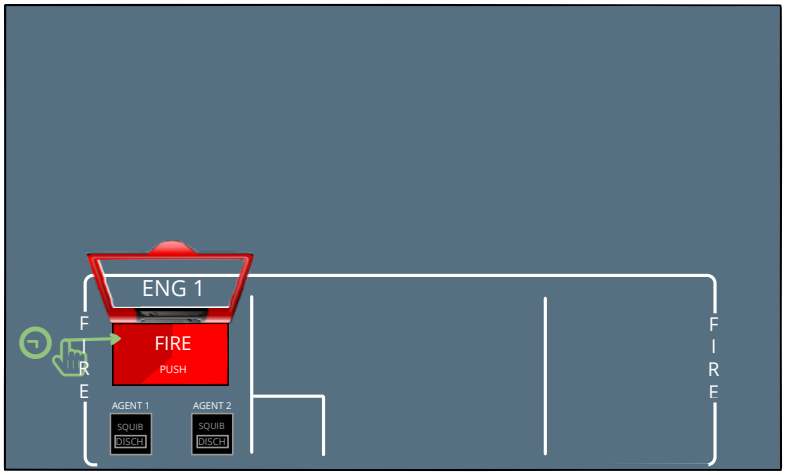

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

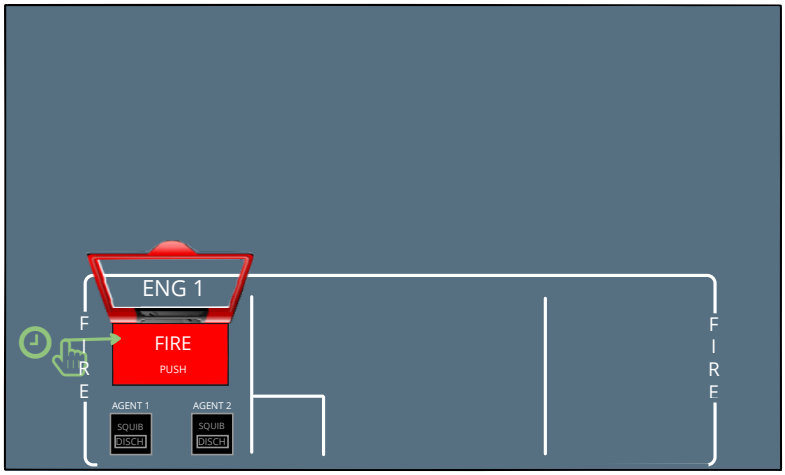

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

Prototype 9

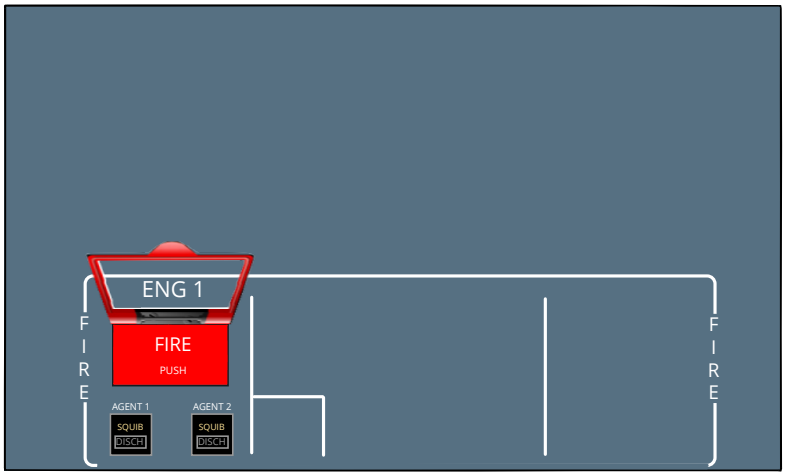

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

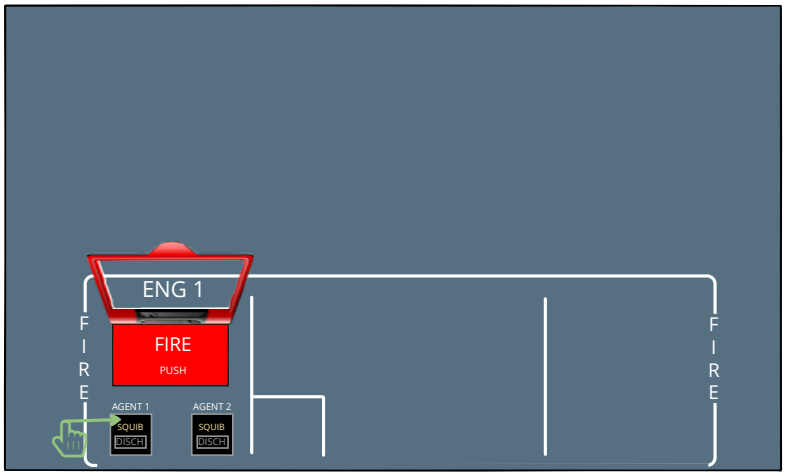

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

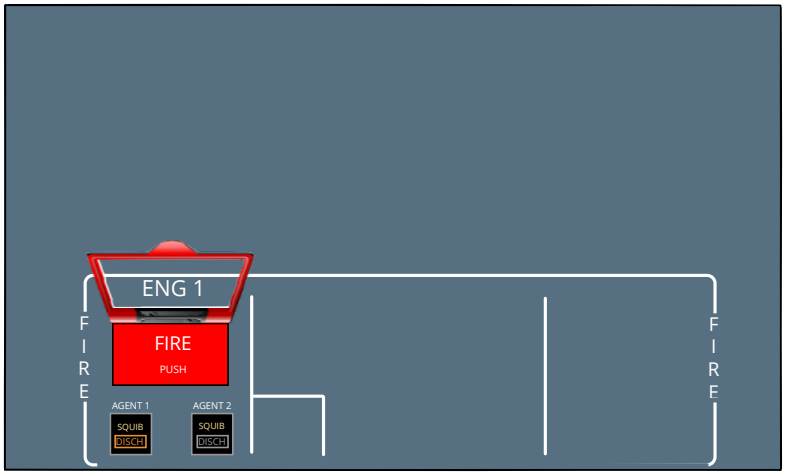

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE


ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
FIRE
FIRE

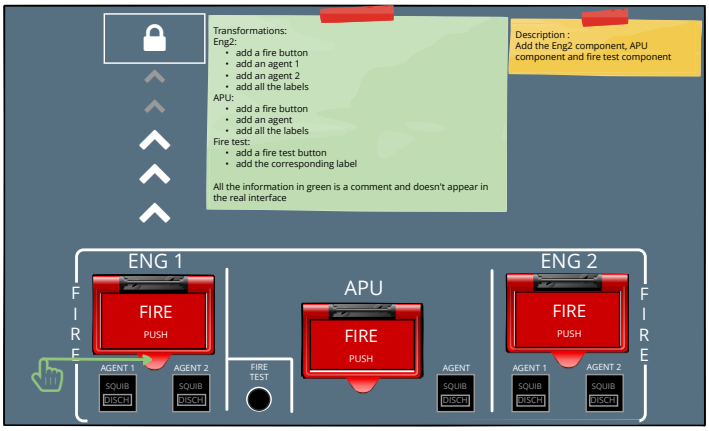
Prototype 9

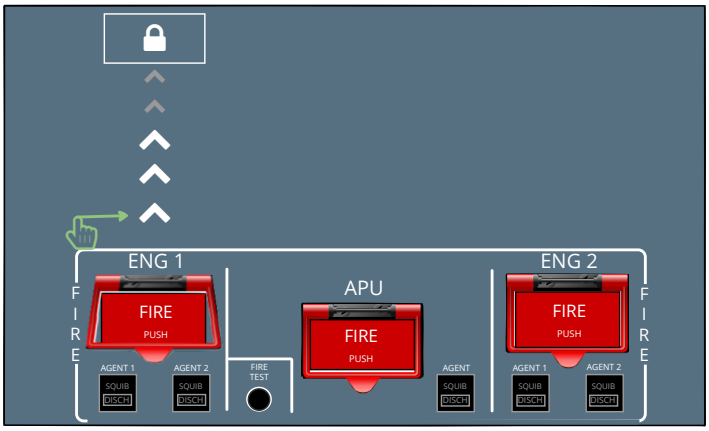

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
SQUIB
DISCH
F I R E
F I R E

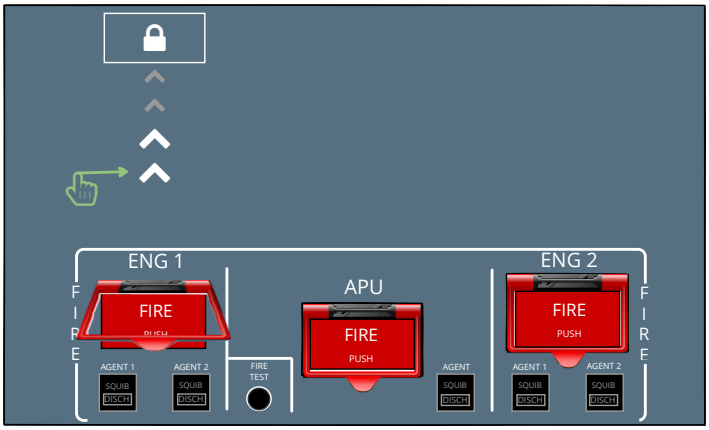

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
SQUIB
F I R E
F I R E

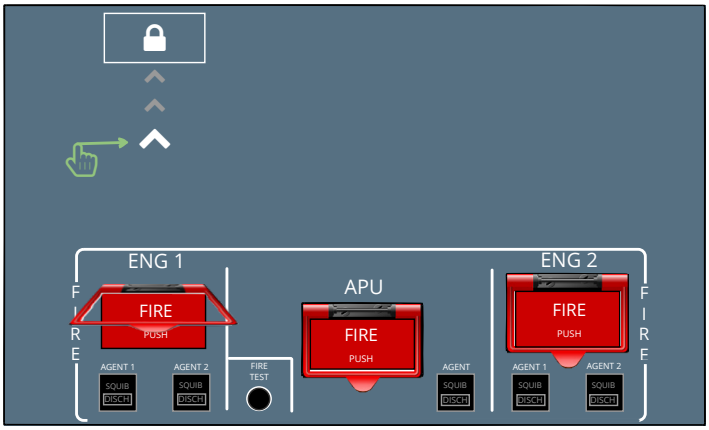

ENG 1
FIRE
PUSH
AGENT 1
AGENT 2
SQUIB
DISCH
SQUIB
DISCH
F I R E
F I R E

Prototype 10

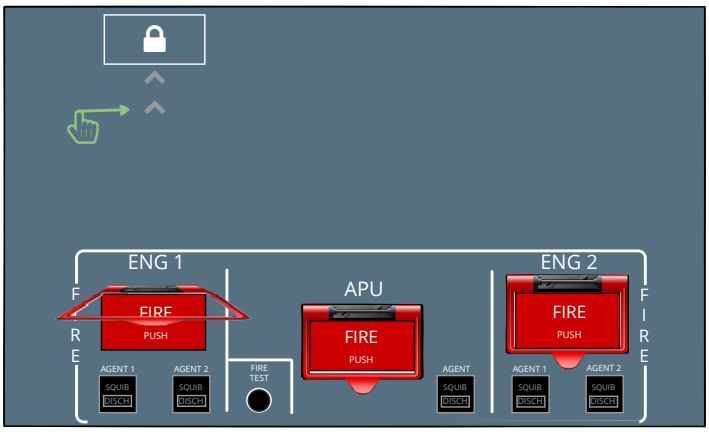


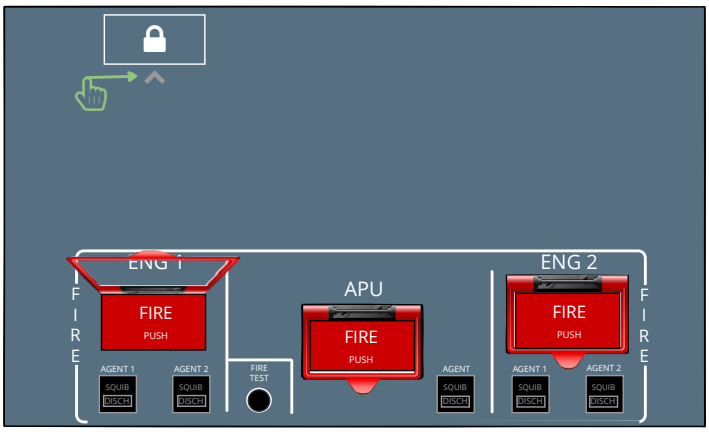


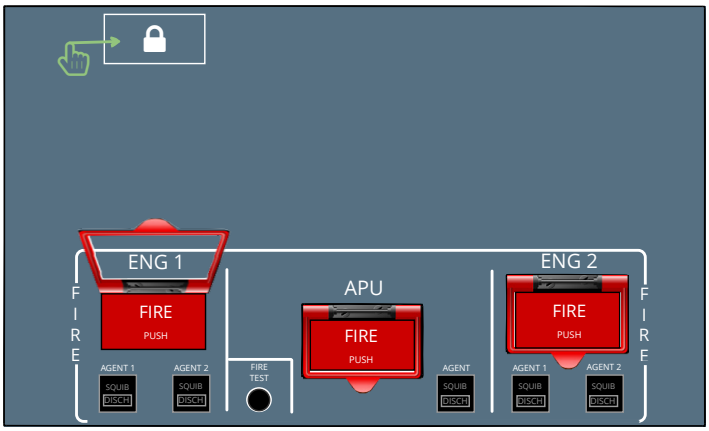


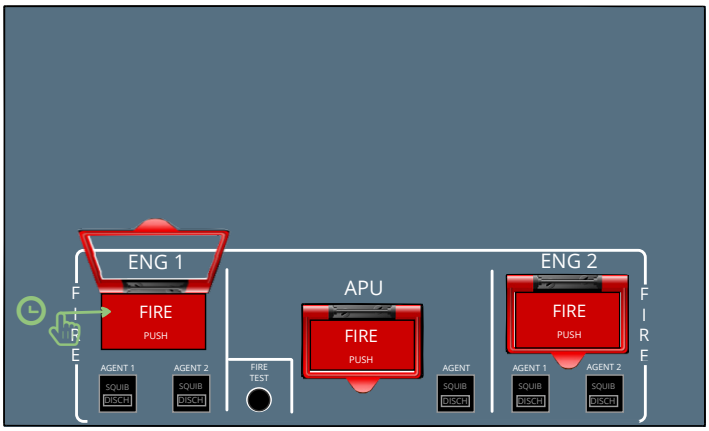


Prototype 10

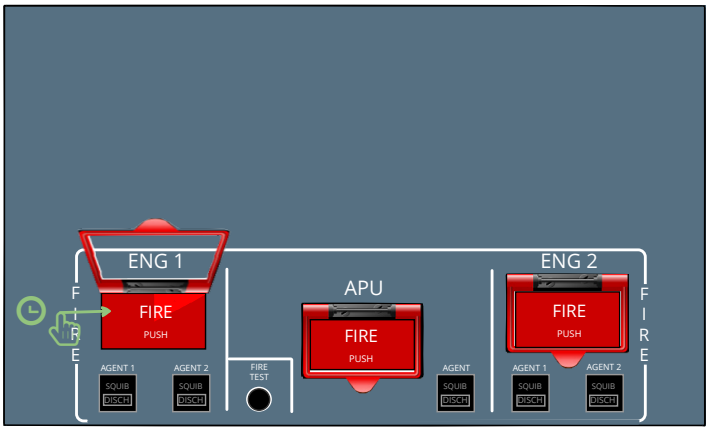


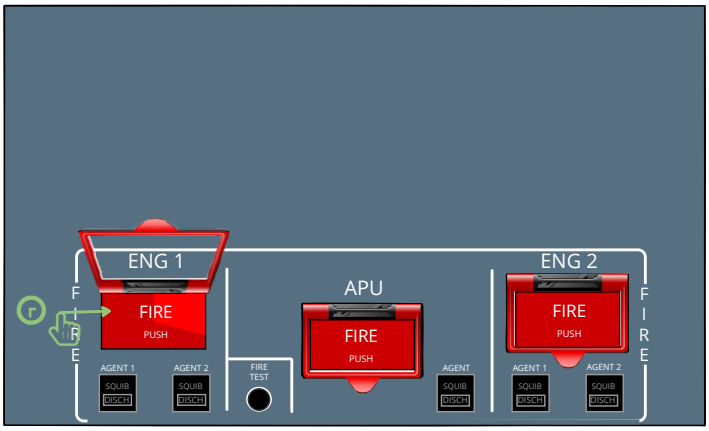


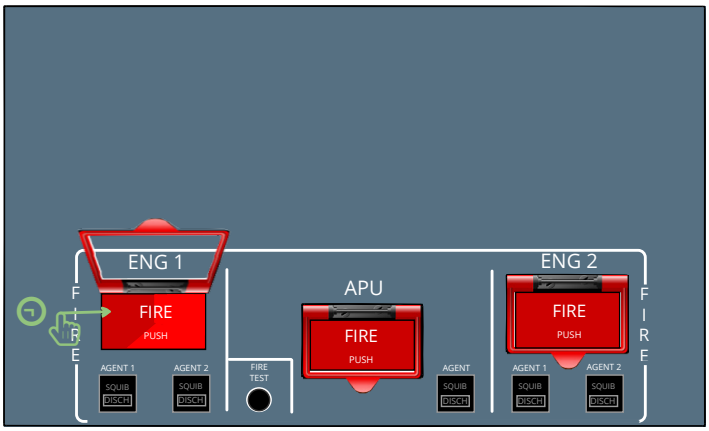


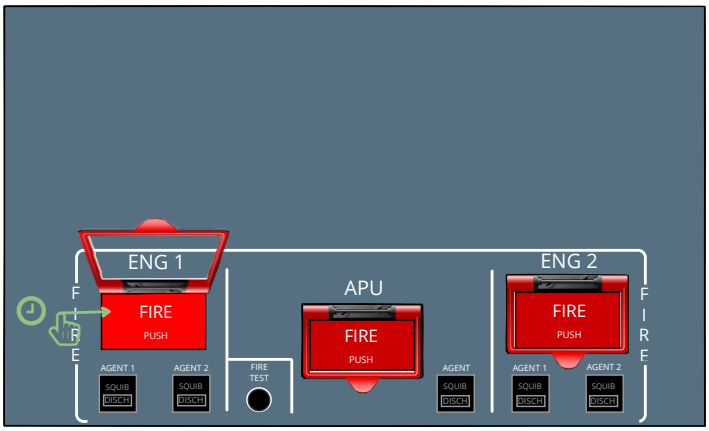


Prototype 10

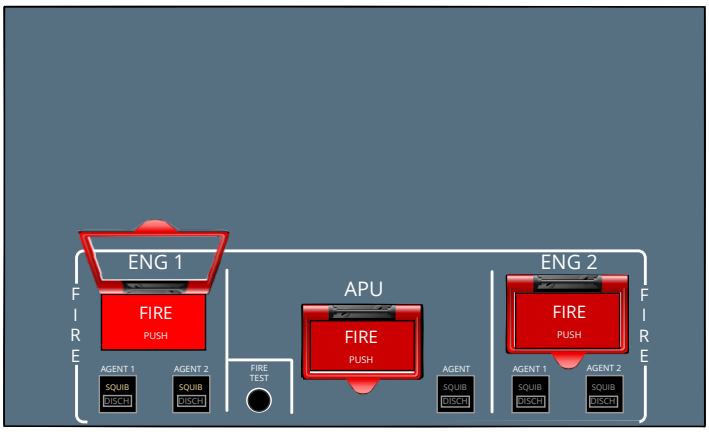


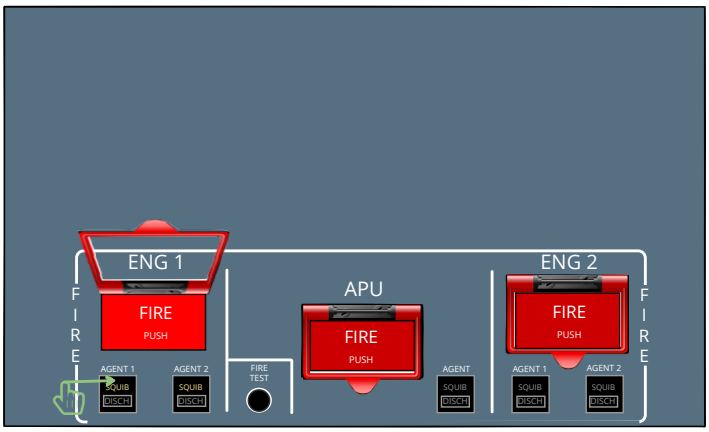


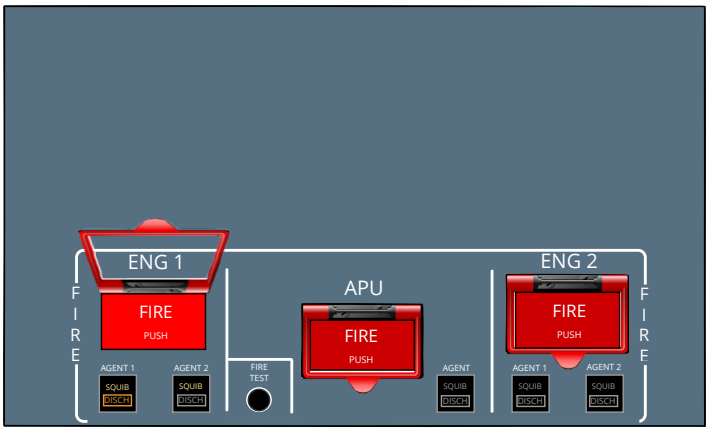




Prototype 10



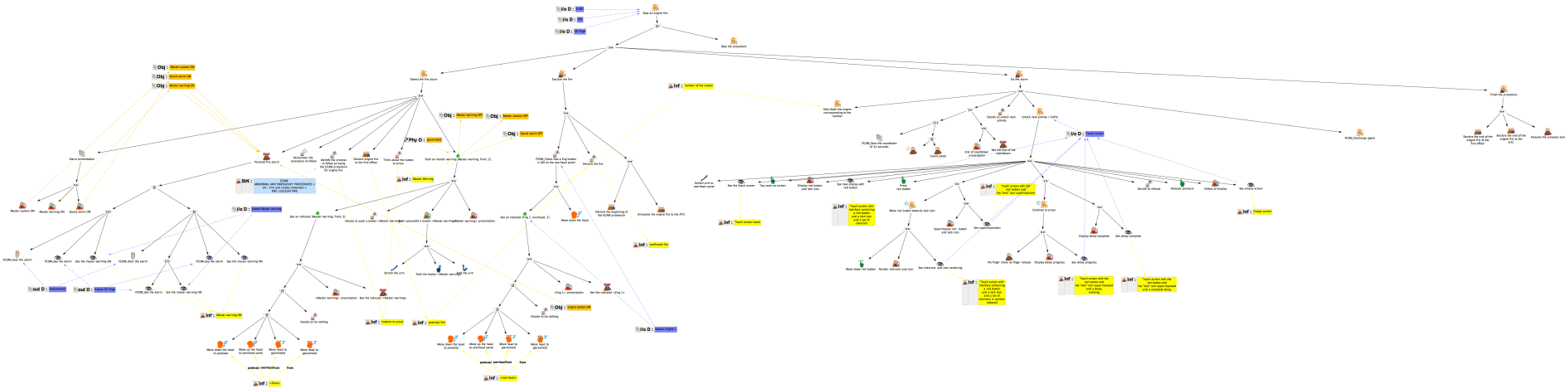




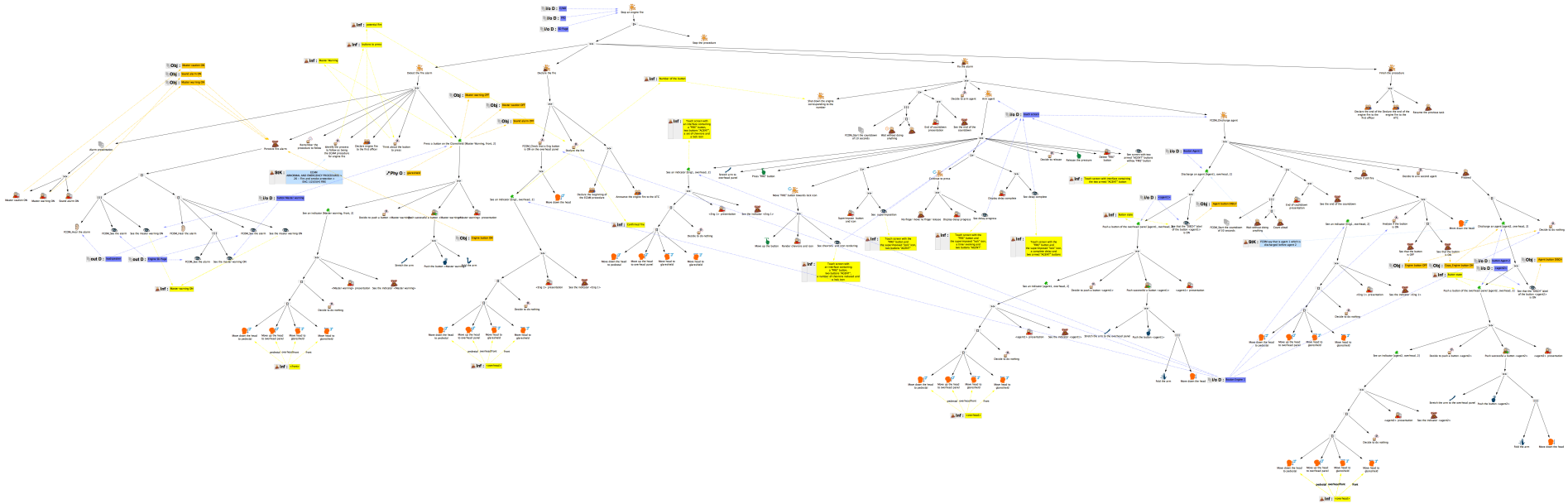
Annexe B

Modèles de tâches des prototypes

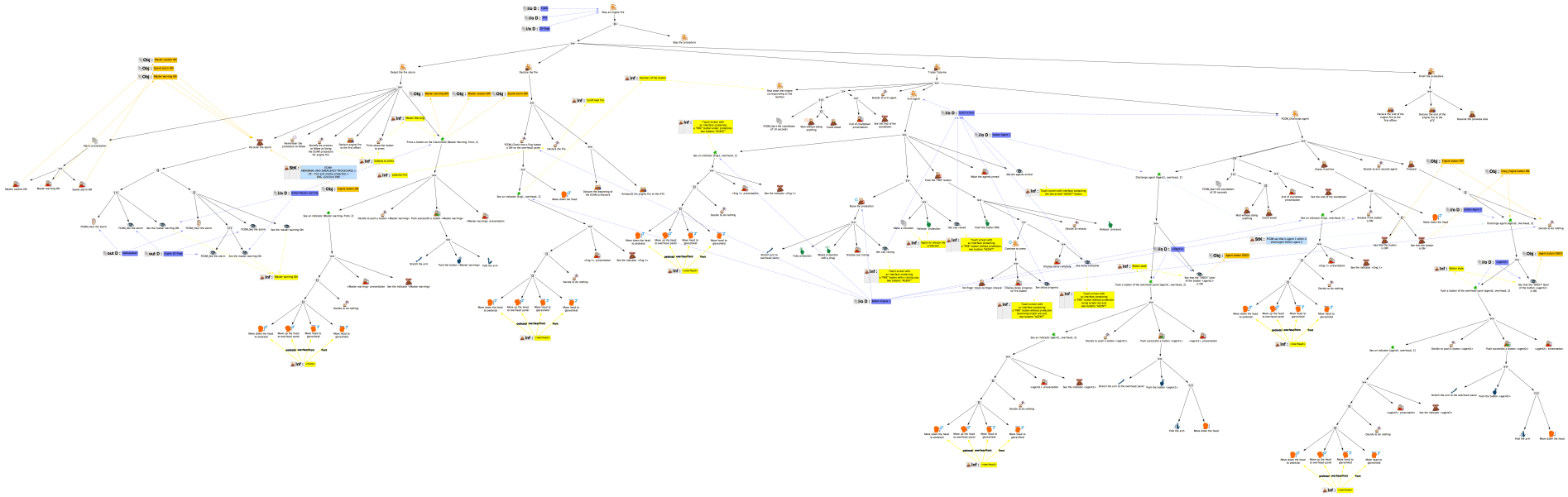
Modèle de tâche du prototype de la GoPro



Modèle de tâche du prototype 7



Modèle de tâche du prototype final



Annexe C

Article scientifique

Similarity as a Design Driver for User Interfaces of Dependable Critical Systems

David Navarre, Philippe Palanque, Arnaud Hamon & Sabrina Della Pasqua

ICS-IRIT, Université Toulouse III, France
{navarre , palanque}@irit.fr

Abstract. Assuring that operators will be able to perform their activities even though the interactive system exhibits failures is one of the main issues to address when designing and implementing interactive systems in safety critical contexts. The zero-defect approaches (usually based on formal methods) aim at guaranteeing that the interactive system will be defect free. While this has been proven a good mean for detecting and removing faults and bugs at development time, natural faults (such as bit-flips due to radiations) are beyond their reach. One of the way to tackle this kind of issue is to propose redundant user interfaces offering multiple ways for the user to perform operations. When one of the interaction mean is failing, the operator can select another functional one. However, to avoid errors and increase learnability, it is important to ensure that the various user interfaces are "similar" at presentation and interaction levels. This paper investigates this relation between dependability and similarity for fault-tolerant interactive systems.

Keywords: UI properties, similarity, dependability, usability, learnability.

1 Introduction

Usability [9] and user experience [7] properties have received (and are still receiving) a lot of attention in the area of Human-Computer Interaction to the extent that they are perceived as the main properties to study and consider while designing interactive systems or while performing research activities in HCI.

Beyond this main stream of research and design, other more marginal approaches have tried to investigate the relationship between these properties and other ones such as security [18], accessibility [19, 16], dependability [3] or privacy [6] (among many others).

Each of these specific domains bring specific issues in order to ensure that the associated properties have been taken into account. Taking into account these properties usually requires identifying and managing trade-off i.e. favoring one property above the other. For instance, adding an undo function to an interactive system will improve usability by make it more efficient for users to recover from errors. However, adding undo functionality to a system increases significantly the number of lines of code and thus the likelihood of bugs. This paper focuses on dependability related issues and how dealing with them might bring additional concerns for the design of user interfaces and

their associated interaction techniques. However, despite this specific focus on one property, similar constraints would apply to other conflicting properties.

Assuring that operators will be able to perform their activities even though the interactive system exhibits failures is one of the main issues to address when designing and implementing interactive systems in safety critical contexts. Exploiting methods, techniques and tools from the dependable computing field [10] can ensure this even though they have not been designed and developed to meet the challenges of interactive systems [4]. Such approaches can be dived into two main categories:

- **The zero-defect approaches** (usually based on formal methods [21]) that aim at guaranteeing that the interactive system will be defect free. While this has been proven a good mean for detecting and removing faults and bugs at development time, natural faults (such as bit-flips due to radiations) are beyond their reach.
- **The fault-tolerant approaches** that promote the use of **redundancy** (multiple versions of the system), **diversity** (the various versions are developed using different means, technologies and providers) and **segregation** (the various versions are integrated in the operational environment by independent means e.g. executed on different computers, using different communication means, …). Segregation ensures that a fault in one of the versions will not induce a fault in another version – usually called common point of failure.

One of the way to apply dependability principles to the user interface of the interactive system is to propose redundant user interfaces offering multiple ways for the user to perform operations. This can be displaying the same information on different screens or offering multiple input devices for triggering the same action. This can also be performed at the interaction technique level as presented in [15] where mouse failures were mitigated by the use of "similar" configurations based on use of multiples keys on the keyboard. However, to avoid user errors (such as capture errors [17]) and increase **learnability**, it is important to ensure that the various user interfaces are "similar" at presentation and interaction levels. This concept of **similarity** has already been used in the field of web engineering [8] but only with a focus of designing new web systems being consistent with legacy non-web systems.

This paper refines the concept of similarity and shows how this concept is relevant at different levels of the architecture of interactive systems. The paper then presents a set of examples from the avionics domain where dependability is a major concern and where development of fault-tolerant mechanisms is a requirement from standardization authorities such as DO 178C standard [1]. These examples present how similarity has been driving the design of multiple user interfaces even though they are as different as hardware only (interaction taking place through knobs and dials) and software mainly using WIMP interaction techniques. Conclusions and discussions for the workshop are presented in the last section.

2 Conflicts and Congruence between Similarity, Diversity and Redundancy in the area of interactive systems

In order to increase resilience to failures, fault-tolerance (i.e. guaranteeing the continuity of service), requires **duplicated user interfaces** for the command and control of a single system. This ends up with **redundant user interfaces** serving the same purpose. If those interfaces are built using the same processes and offer the same interaction techniques, it is possible that a single fault could trigger failures in both user interfaces. This could be the case for instance when using the idea of cloning the UI as proposed by [20]. In order to avoid such common points of failure the redundant user interfaces must ensure **diversity**. Diversity can be guaranteed if the user interfaces have been developed using diverse means such as different programming languages, different notations for describing their specification, executed on top of different operating systems, exploiting different output and input devices, … Such diversity is only efficient if the command and control system offers confinement mechanisms avoiding cascading faults i.e. the failure of one user interface triggering a failure in the duplicated one.

Such fault tolerant basic principles raise **conflicting** design issues when applied to user interfaces. Indeed, diversity requires the user interfaces to be very different in terms of structure, content and in terms of interaction techniques they offer, even though they must guarantee that they support the same tasks and the same goals of the operators [5]. Another aspect is that they must be located in different places in the system i.e. distributed as this is one of the most efficient way of ensuring confinement of faults.

In that context, distribution of user interface does not concern the presentation of complementary information in different contexts (as presented in [12]) but the presentation of redundant information in those contexts.

In terms of design, it is important to be able to assess that the various user interfaces make it possible for the operators to reach their goals (this would be called similarity in terms of **effectiveness**). Beyond that, it is also important to be able to assess the relative complexity and diversity of these interfaces in order to be sure that operations will not be drastically degraded when a redundant user interface has to be used after a failure has occutred on another one. Studying the effective **similarity** (in terms of **efficiency**) at the level of input and output is thus required even though different type of displays and different types of input devices have to be used. This goes beyond the study of similarity at effectiveness level, but both contribute to the usability of the systems. It is important to note that all the other properties mentioned previously are intrinsic or extrinsic propeties of a given interactive system. Similarity is very special as it only has a meaning when two interactive systems are considered (or two different versions of a same interactive system). Such relative properties are usually less studied than absolute properties as the focus of interest is usually to favour a given property of a given system.

3 Examples from the Avionics Domain

The case study presents (in the area of aircraft cockpits) examples of redundant user interfaces. More precisely, we present in the context of the cockpit of the A380 (see Figure 1) aircraft. In this new generation of large civil aircrafts, the cockpit presents display units (that can be considered here as computers screens) of which some of them are offering interaction via a mouse and a keyboard by means of an integrated input device called KCCU (Keyboard Cursor Control Unit). Applications are allocated to the various display unit (DU).

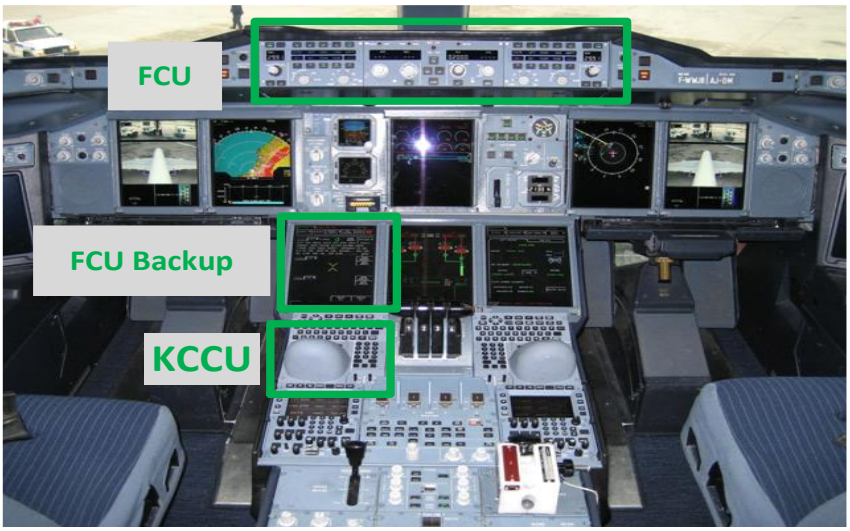


Figure 1. Two possible means to control flight heading within the A380 interactive cockpit, one using the FCU and the other using the FCU Software application and the KCCU

In the A380, two redundant ways of using the autopilot are offered to the pilot in order to change the heading of the aircraft. One is performed using the electronic user interface of the Flight Control Unit (FCU on top of Figure 1)while the other one exploits the graphical user interface of the Flight Control Unit Backup interface and the KCCU (bottom of Figure 1).

3.1 Example one: entering a new value for heading

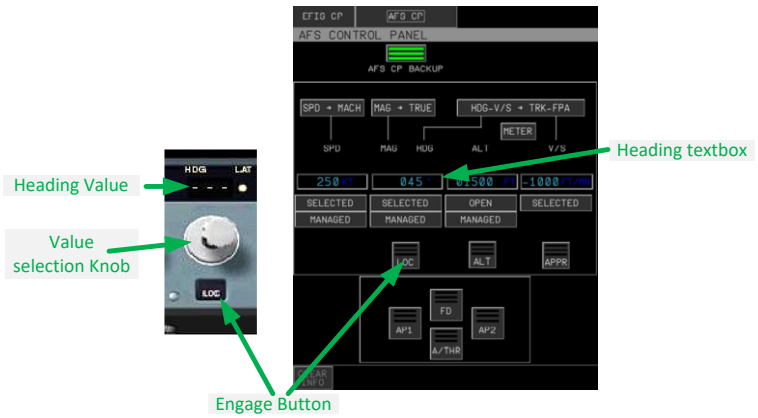


Figure 2. Heading selection.

Figure 2 presents a zoomed view on the two means for entering a new heading of the aircraft. On the left-hand side of the figure, the editing of the heading is performed using a physical knob, which may be turned to set a heading value (this value ranges from 0 to 360). The selected value can be sent to the autopilot (called “engaged”) by pressing the physical LOC push button below the knob. On the right-hand side, the heading is set using the keyboard of the KCCU and engaged by using the KCCU and its manipulator to click on the dedicated software LOC push button.

At a high level of abstraction (i.e. not taking into account the input and output devices), the task of setting a new value for the heading is the same on both user interfaces (they are similar at the effectiveness level). If described at a lower level, the description of these two tasks would be different, as they would require different physical movements from the pilots (they are thus not similar at the effectiveness level as for instance, the pilot would have to execute the FCUS application while the hardware FCU is directly reachable). It is important to note that there are other additional means to perform the same task (for instance controlling directly the aircraft using the sidestick) that are not presented here.

3.2 Example two: entering a set of parameters for the Navigation Display

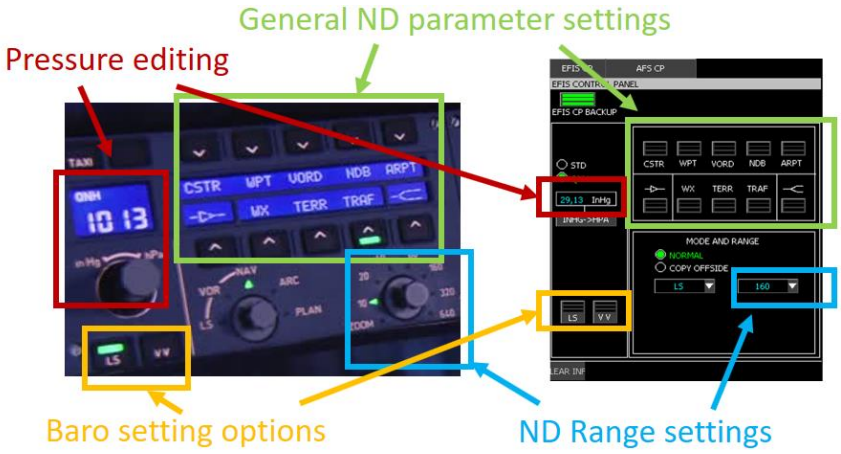


Figure 3. Baro settings and Navigation Display configuration

Figure 3 presents two different means to handle both barometer settings and parameters of the navigation display (ND – pilot ND is the second screen on the left in Figure 1 while first officer ND is the second screen on the right). It illustrates how physical input devices (on the left-hand side of Figure 3) have been transposed into software components (right-hand side of Figure 3) handled using the KCCU (as in the FCUS presented in Figure 2). The general layout of both interface is quite close to that one, but the translation into a software application leads to different design options:

- On the physical interface, the **two barometer settings** options (highlighted in yellow and on the bottom left part of both physical and software interfaces) are handled using two physical labelled push buttons (LS and VV) that are lighted on with a single light when the option is selected. The transposition of these two buttons in the software user interface results is a set of two software buttons that may be highlighted by changing the color of three horizontal lines. In this case, the two design options are quite similar.

- The **General ND parameter settings** (highlighted in green and on the top right part of both physical and software interfaces) are physically handled using physical push buttons without labels associated to labels displayed on a dedicated screen. These buttons behave in the same way as the two previous buttons. The software transposition is similar to the previous one, using both software push buttons and labels, and following the same layout constraints (relative position and size) as the physical interface.

- The **Pressure editing** (highlighted in red and located on the left-hand side of both physical and software interfaces) consists in the editing of a numeric value. The physical and software representations of this function follow two distinct design option. With the physical interface, this value is modified using a physical knob and the edited value is displayed on a dedicated screen while on the software transposition, this editing is performed using a classical text field that embed both editing and display of the value. It is thus possible on

the software UI to use the arrow keys to navigate into the text box and modify one specific digit of the pressure, which is not feasible on the hardware UI.

- The **ND range setting** (highlighted in blue and on the bottom right part of both physical and software interfaces) is performed by selecting a range amongst a finite set of predefined values. In this case, the two design options are quite different too. On the physical interface, the task is performed using a knob that can rotate between the set of values, these values being physically written around the knob (making it visible at any time). The software translation of this interface is made up using a drop down combo box that embed both the display and selection of the value. In this case, the selectable values are only displayed while using the software component.

3.3 Example one: visualization of aircraft pitch and roll

Figure 4 presents two different design of the gyroscope instrument that aims at providing the pilot with information about the position of the aircraft relatively to the horizon (both pitch and roll). At the bottom right-hand side of Figure 4 the cockpit presents the physical analog display of these values. This device is also called the artificial horizon as the information it displays is similar to the view the pilots have when they look outside through the windshield. The software transposition of this instrument (on the left-hand side of Figure 4 – called Primary Flight Display) embeds several other functions such as an altimeter or a speed controller. The graphical layout of the software UI is clearly inspired by the physical one which was, in the early days of aviation only a physical ball emerged in a container filled with liquid.

Figure 4. Physical and software representation of the aircraft gyroscope.

4 Research Directions

While the examples above focus on the presentation and interaction aspects of interactive systems, we are investigating other means to support similarity analysis and assessment also at user level:

- Investigating means of describing past experiences and practice of users to understand the level of familiarity a user may have with a given interaction
- Investigating means of describing tasks (including knowledge, information and objects used to perform the tasks) such as with the HAMSTERS tool [14] to assess similarity of tasks and goals

- Investigating means of describing users' errors (including causes called genotypes and manifestation called phenotypes) in order to identify potential unexpected types of errors that could occur see [2].

In the area of aviation, the design driver for cockpit has been on targeting at similarity of command and displays even though this is not clearly stated. Indeed, looking at training, most airlines propose Cross Crew Qualification programs for pilots [11]. As training is mainly based on tasks execution [13] such a goals and task-based approach is critical and is the only way of designing and evaluating training programs evolutions.

5 Discussions and Conclusion

This paper has presented the similarity property for interactive systems offering redundant ways for the users to enter and perceive information. In order to ensure diversity and segregation (that are required for building dependable interactive systems) the similarity property may be violated. We have shown on the first example that the hardware and the software user interface are similar at the effectiveness level but distinct at interaction level. The following examples have shown bigger gaps in terms of similarity as the use of computing systems and graphical interfaces provides designers and developers with more advanced communication and interaction means. Digital devices are thus more informative and more efficient than the hardware ones. However, they are also less reliable than hardware systems and must not be used if failures are detected [3]. This means that the design and the evaluation of the training program is a complex and expensive activity requiring tools and technique to assess (and explain to trainees) gaps in similarity.

6 References

1. DO-178C / ED-12C, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA and EUROCAE, 2012.
2. Fahssi R., Martinie C., Palanque P. Enhanced Task Modelling for Systematic Identification and Explicit Representation of Human Errors. IFIP TC 13 INTERACT (4) 2015, LNCS, Springer 192-212
3. Fayollas C., Martinie C., Palanque P., Deleris Y., Fabre J-C., Navarre D. An Approach for Assessing the Impact of Dependability on Usability: Application to Interactive Cockpits. Tenth European Dependable Computing Conference (EDCC 2014), IEEE Computer Society: 198-209
4. Fayollas C., Fabre J-C., Palanque P., Cronel M., Navarre D., Deleris Y.A Software-Implemented Fault-Tolerance Approach for Control and Display Systems in Avionics. 20th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2014, 2014. IEEE Computer Society 2014: 21-30
5. Fayollas C., Martinie C., Navarre D., Palanque P., and Fahssi R. Fault-Tolerant User Interfaces for Critical Systems: Duplication, Redundancy and Diversity as New Dimensions of Distributed User Interfaces. *Workshop on Distributed UIs and Multimodal Interaction* (DUI '14), ACM DL, 27-30.
6. Gerber P., Volkamer M., and Renaud K. 2015. Usability versus privacy instead of usable privacy: Google's balancing act between usability and privacy. *SIGCAS Comput. Soc.* 45, 1 (February 2015), 16-21.
7. Hassenzahl M., Platz A., Burmester M., Lehner K. Hedonic and ergonomic quality aspects determine a software's appeal. CHI 2000: 201-208

8. Heil S., Bakaev M., Gaedke M.Measuring and Ensuring Similarity of User Interfaces: The Impact of Web Layout. Web Information Systems Engineering - WISE 2016 - 17th International Conference, Shanghai, China, LNCS 10041, 2016: 252-260
9. International Standard Organization: "ISO 9241-11." Ergonomic requirements for office work with visual display terminals (VDT) – Part 11 Guidance on Usability (1996).
10. Laprie, J. and Randell, B. 2004. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. Dependable Secur. Comput. 1, 1 (Jan. 2004), 11-33
11. Lufthansa: Cross Crew Qualification courses, internet https://www.lufthansa-flight-training.com/documents/10156/5537743/Cross+Crew+Qualification+Course+%28CCQ%29 retrieved November 2nd, 2017.
12. Martinie C., Navarre D., Palanque P. A multi-formalism approach for model-based dynamic distribution of user interfaces of critical interactive systems. Int. J. Hum.-Comput. Stud. 72(1): 77-99 (2014).
13. Martinie C., Palanque P., Navarre D., Winckler M., Poupart E. Model-based training an approach supporting operability of critical interactive systems. EICS 2011, ACM DL, 53-62
14. Martinie C., Palanque P., Ragosta M., Fahssi R. Extending procedural task models by systematic explicit integration of objects, knowledge and information. ECCE 2013, ACM DL, 23:1-23:10
15. Navarre D., Palanque P., Basnyat S. A Formal Approach for User Interaction Reconfiguration of Safety Critical Interactive Systems. SAFECOMP 2008: 373-386
16. Petrie H. and Kheir O.: The relationship between accessibility and usability of websites. SIGCHI Conference on Human Factors in Computing Systems (CHI '07). ACM, 397-406 (2007).
17. Reason J., "Human Error", 1990, Cambridge University Press.
18. Sasse M. A., Karat C.-M., and Maxion R.: Designing and evaluating usable security and privacy technology. 5th Symposium on Usable Privacy and Security (SOUPS '09). ACM, (2009).
19. Section 508: The Road to Accessibility. Available at: http://www.section508.gov/
20. Villanueva,P. G., Tesoriero, R., Gallud, J. A. 2013. Distributing web components in a display ecosystem using Proxywork. 27th BCS HCI Conference (BCS-HCI '13), 2013, British Computer Society.
21. Weyers B., Bowen J., Dix A., Palanque P. The Handbook of Formal Methods in Human-Computer Interaction. Springer International Publishing 2017, ISBN 978-3-319-51837-4